

# zigbee alliance

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11

## Zigbee Cluster Library Specification

Chapter Document 14-0125

Cluster Library 07-5123 Revision 7

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

12

Copyright © 2007-2018 by the Zigbee Alliance.  
<http://www.zigbee.org>  
All rights reserved.

Permission is granted to members of the Zigbee Alliance to reproduce this document for their own use or the use of other Zigbee Alliance members only, provided this notice is included. All other rights reserved. Duplication for sale, or for commercial or for-profit use is strictly prohibited without the prior written consent of the Zigbee Alliance.

## 13 **Notice of Use and Disclosure**

---

14 Copyright © Zigbee Alliance, Inc. (1996-2018). All rights Reserved. This information within this document is the  
15 property of the Zigbee Alliance and its use and disclosure are restricted.

16 Elements of Zigbee Alliance specifications may be subject to third party intellectual property rights, including without  
17 limitation, patent, copyright or trademark rights (such a third party may or may not be a member of Zigbee). Zigbee  
18 is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such  
19 third party intellectual property rights.

20 No right to use any Zigbee name, logo or trademark is conferred herein. Use of any Zigbee name, logo or trademark  
21 requires membership in the Zigbee Alliance and compliance with the Zigbee Logo and Trademark Policy and related  
22 Zigbee policies.

23 This document and the information contained herein are provided on an “AS IS” basis and Zigbee DISCLAIMS ALL  
24 WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT  
25 THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES  
26 (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT,  
27 COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY,  
28 FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN NO EVENT WILL ZIGBEE  
29 BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION  
30 OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL,  
31 PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN  
32 CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF  
33 ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. All Company, brand and product names may  
34 be trademarks that are the sole property of their respective owners.

35 The above notice and this paragraph must be included on all copies of this document that are made.

## 36 **Participants**

---

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

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

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

39

## 40 Document Control

---

- 41 The Zigbee Cluster Library is made of individual chapters such as this one. See Chapter 1 for the list of all chapters.  
42 References between chapters are made using a *X.Y* notation where *X* is the chapter and *Y* is the sub-section within that  
43 chapter. References to external documents are contained in Chapter 1 and are made using [*Rn*] notation.
- 44 An update to any of these chapters will be reflected in an update to the source document list below.

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

45

## Document History

Rev	Date	Comments
00	11-Jul-2007	Document created
01	19-Oct-2007	First release
02	29-May-2008	<p>Added Commissioning Cluster from 064699r12.</p> <ul style="list-style-type: none"> <li>Added material from annex of CBA Profile 053516r10</li> <li>Structured types (arrays etc) and structured R/W commands</li> <li>Input / Output / Value clusters (Basic)</li> <li>Input / Output / Value clusters (BACnet Regular &amp; Extended)</li> <li>Generic Tunnel cluster</li> <li>BACnet Protocol Tunnel cluster</li> </ul> <p>Made changes to the Color Control cluster re. CCB 870</p> <ul style="list-style-type: none"> <li>Added x,y control according to CIE 1931 Color Space</li> </ul> <p>Added long data types (as required by SE profile 075356r12 etc)</p> <ul style="list-style-type: none"> <li>40-64bit integers etc, long strings</li> </ul> <p>Made changes to time cluster (as required by CCBs 890, 914)</p> <ul style="list-style-type: none"> <li>Added time zone &amp; DST + UTCtime type</li> </ul> <p>Made minor changes as requested by the following CCBs</p> <ul style="list-style-type: none"> <li>627, 714, 781, 853, 854, 867, 878, 879, 880, 881, 883, 893, 897, 898, 919, 958</li> </ul>
03	18-Sep-2009	<p>The following changes were made to the Editor's Copy of the ZCL, 095254r00.</p> <p>Made change to the Basic cluster, re CCB comment #606</p> <ul style="list-style-type: none"> <li>Added optional attribute <i>DisableLocalConfig</i>.</li> </ul> <p>Updated Pressure Measurement cluster re CCB comment #961</p> <ul style="list-style-type: none"> <li>Added extra attributes to allow wider range of pressure.</li> </ul> <p>Updated Color Control cluster re CCB comment #1006</p> <ul style="list-style-type: none"> <li>Clarification of stop commands, color mode switching etc.</li> </ul> <p>Made changes to RSSI Location cluster, re CCB comment #1053</p> <ul style="list-style-type: none"> <li>Added mechanism for centralized location.</li> </ul> <p>Made change to Generic Tunnel cluster, re CCB comment #1068</p> <ul style="list-style-type: none"> <li>Added extra fields to Match Protocol Address Response Command</li> </ul>
	24-Dec-2009	<p>Made minor changes and clarifications re the following CCBs</p> <ul style="list-style-type: none"> <li>960, 1001, 1004, 1061, 1097.</li> <li>Added Door Lock cluster.</li> </ul> <p>Updated Occupancy Sensor re CCB comments 1092, 1093, 1094</p>
04	2010	<p>CCB 1174: Fixed references</p> <p>CCB 1176: Added new status codes</p> <p>CCB 1202: Corrected default value in thermostat cluster</p>
	Apr-2012	CCB 1381: Default Response clarification

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

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

47

48

# 49 TABLE OF CONTENTS

50	<b>Zigbee Cluster Library Specification .....</b>	<b>1</b>
51	Notice of Use and Disclosure.....	2
52	Participants.....	3
53	Document Control .....	4
54	Document History .....	5
55	Table of Contents .....	8
56	List of Figures.....	19
57	List of Tables .....	28
58	<b>Chapter 1 Introduction .....</b>	<b>1-1</b>
59	1.1 Scope and Purpose .....	1-1
60	1.2 Acronyms and Abbreviations .....	1-1
61	1.3 Definitions.....	1-4
62	1.4 Conformance Levels .....	1-5
63	1.5 References .....	1-5
64	<b>1.5.1</b> Zigbee Alliance Documents.....	1-5
65	<b>1.5.2</b> International Standards Documents.....	1-5
66	<b>1.5.3</b> National Standards Documents .....	1-6
67	<b>1.5.4</b> IEEE Documents.....	1-6
68	<b>1.5.5</b> ASHRAE Documents.....	1-6
69	<b>1.5.6</b> Health Care Documents.....	1-6
70	<b>1.5.7</b> Other Documents .....	1-7
71	1.6 Conventions.....	1-7
72	<b>1.6.1</b> Enumerations and Reserved Values .....	1-7
73	<b>1.6.2</b> Reserved Bit Fields .....	1-7
74	<b>1.6.3</b> Number Format.....	1-7
75	1.7 Testing, Validation and Certification .....	1-8
76	<b>Chapter 2 Foundation.....</b>	<b>2-1</b>
77	2.1 Scope and Purpose .....	2-1
78	2.2 Cluster Library Overview .....	2-1
79	<b>2.2.1</b> Architecture and Data Model.....	2-1
80	<b>2.2.2</b> Client/Server Model.....	2-2
81	2.3 Functional Description .....	2-3
82	<b>2.3.1</b> Transmission.....	2-3
83	<b>2.3.2</b> Reception.....	2-3
84	<b>2.3.3</b> Manufacturer Specific Extensions .....	2-4
85	<b>2.3.4</b> Attributes.....	2-5
86	<b>2.3.5</b> Persistent Data .....	2-7
87	2.4 Command Frame Formats .....	2-7
88	<b>2.4.1</b> General ZCL Frame Format .....	2-7
89	2.5 General Command Frames .....	2-9
90	<b>2.5.1</b> Read Attributes Command .....	2-10
91	<b>2.5.2</b> Read Attributes Response Command.....	2-11
92	<b>2.5.3</b> Write Attributes Command.....	2-13
93	<b>2.5.4</b> Write Attributes Undivided Command.....	2-15
94	<b>2.5.5</b> Write Attributes Response Command .....	2-15
95	<b>2.5.6</b> Write Attributes No Response Command.....	2-16
96	<b>2.5.7</b> Configure Reporting Command.....	2-17
97	<b>2.5.8</b> Configure Reporting Response Command .....	2-20



98	<b>2.5.9</b>	Read Reporting Configuration Command .....	2-21
99	<b>2.5.10</b>	Read Reporting Configuration Response Command.....	2-22
100	<b>2.5.11</b>	Report Attributes Command.....	2-24
101	<b>2.5.12</b>	Default Response Command .....	2-27
102	<b>2.5.13</b>	Discover Attributes Command.....	2-28
103	<b>2.5.14</b>	Discover Attributes Response Command .....	2-29
104	<b>2.5.15</b>	Read Attributes Structured Command.....	2-30
105	<b>2.5.16</b>	Write Attributes Structured Command.....	2-31
106	<b>2.5.17</b>	Write Attributes Structured Response Command .....	2-34
107	<b>2.5.18</b>	Discover Commands Received Command .....	2-35
108	<b>2.5.19</b>	Discover Commands Received Response.....	2-36
109	<b>2.5.20</b>	Discover Commands Generated Command.....	2-37
110	<b>2.5.21</b>	Discover Commands Generated Response .....	2-38
111	<b>2.5.22</b>	Discover Attributes Extended Command .....	2-38
112	<b>2.5.23</b>	Discover Attributes Extended Response Command.....	2-39
113	2.6	Addressing, Types and Enumerations .....	2-41
114	<b>2.6.1</b>	Addressing.....	2-41
115	<b>2.6.2</b>	Data Types.....	2-43
116	<b>2.6.3</b>	Status Enumerations.....	2-52
117	Chapter 3	General.....	3-1
118	3.1	General Description .....	3-1
119	<b>3.1.1</b>	Introduction .....	3-1
120	<b>3.1.2</b>	Cluster List .....	3-1
121	3.2	Basic .....	3-6
122	<b>3.2.1</b>	Overview .....	3-6
123	<b>3.2.2</b>	Server .....	3-6
124	<b>3.2.3</b>	Client.....	3-17
125	3.3	Power Configuration.....	3-17
126	<b>3.3.1</b>	Overview .....	3-17
127	<b>3.3.2</b>	Server .....	3-17
128	<b>3.3.3</b>	Client.....	3-26
129	3.4	Device Temperature Configuration.....	3-26
130	<b>3.4.1</b>	Overview .....	3-26
131	<b>3.4.2</b>	Server .....	3-27
132	<b>3.4.3</b>	Client.....	3-29
133	3.5	Identify.....	3-29
134	<b>3.5.1</b>	Overview .....	3-29
135	<b>3.5.2</b>	Server .....	3-30
136	<b>3.5.3</b>	Client.....	3-33
137	3.6	Groups.....	3-33
138	<b>3.6.1</b>	Overview .....	3-33
139	<b>3.6.2</b>	Server .....	3-35
140	<b>3.6.3</b>	Client.....	3-42
141	3.7	Scenes .....	3-42
142	<b>3.7.1</b>	Overview .....	3-42
143	<b>3.7.2</b>	Server .....	3-42
144	<b>3.7.3</b>	Client.....	3-56
145	3.8	On/Off.....	3-56
146	<b>3.8.1</b>	Overview .....	3-56
147	<b>3.8.2</b>	Server .....	3-56
148	<b>3.8.3</b>	Client.....	3-63
149	3.9	On/Off Switch Configuration .....	3-64
150	<b>3.9.1</b>	Overview .....	3-64
151	<b>3.9.2</b>	Server .....	3-64
152	<b>3.9.3</b>	Client.....	3-66

153	3.10 Level .....	3-66
154	<b>3.10.1</b> Overview .....	3-66
155	<b>3.10.2</b> Server .....	3-67
156	<b>3.10.3</b> Client.....	3-76
157	3.11 Alarms.....	3-76
158	<b>3.11.1</b> Overview .....	3-76
159	<b>3.11.2</b> Server .....	3-77
160	<b>3.11.3</b> Client.....	3-80
161	3.12 Time.....	3-80
162	<b>3.12.1</b> Overview .....	3-80
163	<b>3.12.2</b> Server .....	3-81
164	<b>3.12.3</b> Client.....	3-84
165	3.13 RSSI Location.....	3-84
166	<b>3.13.1</b> Overview .....	3-84
167	<b>3.13.2</b> Server .....	3-86
168	<b>3.13.3</b> Client.....	3-99
169	3.14 Input, Output and Value Clusters.....	3-99
170	<b>3.14.1</b> Overview .....	3-99
171	<b>3.14.2</b> Analog Input (Basic) .....	3-99
172	<b>3.14.3</b> Analog Output (Basic).....	3-101
173	<b>3.14.4</b> Analog Value (Basic).....	3-102
174	<b>3.14.5</b> Binary Input (Basic).....	3-104
175	<b>3.14.6</b> Binary Output (Basic) .....	3-105
176	<b>3.14.7</b> Binary Value (Basic).....	3-107
177	<b>3.14.8</b> Multistate Input (Basic).....	3-108
178	<b>3.14.9</b> Multistate Output (Basic) .....	3-110
179	<b>3.14.10</b> Multistate Value (Basic).....	3-111
180	<b>3.14.11</b> Attribute Descriptions .....	3-112
181	3.15 Diagnostics .....	3-149
182	<b>3.15.1</b> Overview .....	3-149
183	<b>3.15.2</b> Server .....	3-149
184	<b>3.15.3</b> Client.....	3-154
185	3.16 Poll Control .....	3-154
186	<b>3.16.1</b> Overview .....	3-154
187	<b>3.16.2</b> Terminology .....	3-155
188	<b>3.16.3</b> Commissioning Process .....	3-155
189	<b>3.16.4</b> Server .....	3-155
190	<b>3.16.5</b> Client.....	3-158
191	<b>3.16.6</b> Poll Control Cluster Sequence Diagram.....	3-161
192	3.17 Power Profile.....	3-163
193	<b>3.17.1</b> Overview .....	3-163
194	<b>3.17.2</b> References .....	3-164
195	<b>3.17.3</b> General Description .....	3-164
196	<b>3.17.4</b> Server Attributes .....	3-165
197	<b>3.17.5</b> Server Commands Received.....	3-166
198	<b>3.17.6</b> Server Commands Generated .....	3-174
199	<b>3.17.7</b> Client Attributes.....	3-185
200	<b>3.17.8</b> Client Commands Received.....	3-185
201	<b>3.17.9</b> Client Commands Generated.....	3-185
202	<b>3.17.10</b> Example of Device Interactions Using the Power Profile (Informative Section) .....	3-185
203	3.18 Meter Identification.....	3-188
204	<b>3.18.1</b> Overview .....	3-188
205	<b>3.18.2</b> Server .....	3-189
206	<b>3.18.3</b> Client.....	3-192
207	3.19 Level Control for Lighting .....	3-192
208	<b>3.19.1</b> Overview .....	3-192

209	<b>3.19.2</b>	Server .....	3-193
210	<b>3.19.3</b>	Client.....	3-195
211	<b>3.19.4</b>	State Change Table for Lighting .....	3-195
212	3.20	Pulse Width Modulation.....	3-197
213	<b>3.20.1</b>	Overview .....	3-197
214	<b>3.20.2</b>	Server .....	3-197
215	<b>3.20.3</b>	Client.....	3-198
216	Chapter 4	Measurement and Sensing.....	4-1
217	4.1	General Description .....	4-1
218	<b>4.1.1</b>	Introduction .....	4-1
219	<b>4.1.2</b>	Cluster List .....	4-1
220	<b>4.1.3</b>	Measured Value .....	4-4
221	4.2	Illuminance Measurement .....	4-5
222	<b>4.2.1</b>	Overview .....	4-5
223	<b>4.2.2</b>	Server .....	4-5
224	<b>4.2.3</b>	Client.....	4-7
225	4.3	Illuminance Level Sensing .....	4-7
226	<b>4.3.1</b>	Overview .....	4-7
227	<b>4.3.2</b>	Server .....	4-7
228	<b>4.3.3</b>	Client.....	4-9
229	4.4	Temperature Measurement.....	4-10
230	<b>4.4.1</b>	Overview .....	4-10
231	<b>4.4.2</b>	Server .....	4-10
232	<b>4.4.3</b>	Client.....	4-12
233	4.5	Pressure Measurement.....	4-12
234	<b>4.5.1</b>	Overview .....	4-12
235	<b>4.5.2</b>	Server .....	4-12
236	<b>4.5.3</b>	Client.....	4-15
237	4.6	Flow Measurement.....	4-15
238	<b>4.6.1</b>	Overview .....	4-15
239	<b>4.6.2</b>	Server .....	4-16
240	<b>4.6.3</b>	Client.....	4-17
241	4.7	Water Content Measurement .....	4-17
242	<b>4.7.1</b>	Overview .....	4-17
243	<b>4.7.2</b>	Server .....	4-18
244	<b>4.7.3</b>	Client.....	4-19
245	4.8	Occupancy Sensing.....	4-19
246	<b>4.8.1</b>	Overview .....	4-19
247	<b>4.8.2</b>	Server .....	4-19
248	<b>4.8.3</b>	Client.....	4-23
249	4.9	Electrical Measurement.....	4-23
250	<b>4.9.1</b>	Overview .....	4-23
251	<b>4.9.2</b>	Server .....	4-24
252	4.10	Electrical Conductivity Measurement.....	4-46
253	<b>4.10.1</b>	Overview .....	4-46
254	<b>4.10.2</b>	Server .....	4-46
255	<b>4.10.3</b>	Client.....	4-47
256	4.11	pH Measurement.....	4-47
257	<b>4.11.1</b>	Overview .....	4-47
258	<b>4.11.2</b>	Server .....	4-48
259	<b>4.11.3</b>	Client.....	4-49
260	4.12	Wind Speed Measurement.....	4-49
261	<b>4.12.1</b>	Overview .....	4-49
262	<b>4.12.2</b>	Server .....	4-49
263	<b>4.12.3</b>	Client.....	4-50

264	4.13 Concentration Measurement .....	4-50
265	<b>4.13.1</b> Overview .....	4-50
266	<b>4.13.2</b> Server .....	4-53
267	<b>4.13.3</b> Client.....	4-53
268	Chapter 5 Lighting.....	5-1
269	5.1 General Description .....	5-1
270	<b>5.1.1</b> Introduction .....	5-1
271	<b>5.1.2</b> Terms .....	5-1
272	<b>5.1.3</b> Cluster List .....	5-1
273	5.2 Color Control Cluster.....	5-2
274	<b>5.2.1</b> Overview .....	5-2
275	<b>5.2.2</b> Server .....	5-3
276	<b>5.2.3</b> Client.....	5-35
277	5.3 Ballast Configuration Cluster .....	5-35
278	<b>5.3.1</b> Overview .....	5-35
279	<b>5.3.2</b> Server .....	5-35
280	<b>5.3.3</b> Client.....	5-40
281	<b>5.3.4</b> The Dimming Light Curve .....	5-40
282	Chapter 6 HVAC .....	6-1
283	6.1 General Description .....	6-1
284	<b>6.1.1</b> Introduction .....	6-1
285	<b>6.1.2</b> Terms .....	6-1
286	<b>6.1.3</b> Cluster List .....	6-1
287	6.2 Pump Configuration and Control.....	6-3
288	<b>6.2.1</b> Overview .....	6-3
289	<b>6.2.2</b> Server .....	6-3
290	<b>6.2.3</b> Client.....	6-13
291	6.3 Thermostat.....	6-13
292	<b>6.3.1</b> Overview .....	6-13
293	<b>6.3.2</b> Server .....	6-14
294	<b>6.3.3</b> Client.....	6-37
295	6.4 Fan Control.....	6-37
296	<b>6.4.1</b> Overview .....	6-37
297	<b>6.4.2</b> Server .....	6-37
298	<b>6.4.3</b> Client.....	6-38
299	6.5 Dehumidification Control.....	6-39
300	<b>6.5.1</b> Overview .....	6-39
301	<b>6.5.2</b> Server .....	6-39
302	<b>6.5.3</b> Client.....	6-41
303	6.6 Thermostat User Interface Configuration.....	6-42
304	<b>6.6.1</b> Overview .....	6-42
305	<b>6.6.2</b> Server .....	6-42
306	<b>6.6.3</b> Client.....	6-45
307	Chapter 7 Closures.....	7-1
308	7.1 General Description .....	7-1
309	<b>7.1.1</b> Introduction .....	7-1
310	<b>7.1.2</b> Cluster List .....	7-1
311	7.2 Shade Configuration .....	7-2
312	<b>7.2.1</b> Overview .....	7-2
313	<b>7.2.2</b> Server .....	7-3
314	<b>7.2.3</b> Client.....	7-5
315	7.3 Door Lock .....	7-5
316	<b>7.3.1</b> Overview .....	7-5

317	<b>7.3.2</b>	Server .....	7-5
318	<b>7.3.3</b>	Client.....	7-49
319	7.4	Window Covering.....	7-49
320	<b>7.4.1</b>	Overview .....	7-49
321	<b>7.4.2</b>	Server .....	7-49
322	<b>7.4.3</b>	Client.....	7-58
323	Chapter 8	Security and Safety .....	8-1
324	8.1	General Description .....	8-1
325	<b>8.1.1</b>	Introduction .....	8-1
326	<b>8.1.2</b>	Cluster List .....	8-1
327	8.2	IAS Zone.....	8-2
328	<b>8.2.1</b>	Overview .....	8-2
329	<b>8.2.2</b>	Server .....	8-3
330	<b>8.2.3</b>	Client.....	8-13
331	8.3	IAS ACE .....	8-13
332	<b>8.3.1</b>	Overview .....	8-13
333	<b>8.3.2</b>	Server .....	8-13
334	<b>8.3.3</b>	Client.....	8-27
335	8.4	IAS WD.....	8-27
336	<b>8.4.1</b>	Overview .....	8-27
337	<b>8.4.2</b>	Server .....	8-28
338	<b>8.4.3</b>	Client.....	8-32
339	Chapter 9	Protocol Interfaces .....	9-1
340	9.1	General Description .....	9-1
341	<b>9.1.1</b>	Introduction .....	9-1
342	<b>9.1.2</b>	Cluster List.....	9-1
343	9.2	Generic Tunnel .....	9-3
344	<b>9.2.1</b>	Overview .....	9-3
345	<b>9.2.2</b>	Server .....	9-3
346	<b>9.2.3</b>	Client.....	9-6
347	9.3	BACnet Protocol Tunnel.....	9-6
348	<b>9.3.1</b>	Overview .....	9-6
349	<b>9.3.2</b>	Server .....	9-6
350	<b>9.3.3</b>	Client.....	9-7
351	9.4	BACnet Input, Output and Value Clusters .....	9-8
352	<b>9.4.1</b>	Overview .....	9-8
353	<b>9.4.2</b>	Analog Input (BACnet Regular).....	9-8
354	<b>9.4.3</b>	Analog Input (BACnet Extended).....	9-9
355	<b>9.4.4</b>	Analog Output (BACnet Regular).....	9-11
356	<b>9.4.5</b>	Analog Output (BACnet Extended).....	9-12
357	<b>9.4.6</b>	Analog Value (BACnet Regular).....	9-13
358	<b>9.4.7</b>	Analog Value (BACnet Extended).....	9-14
359	<b>9.4.8</b>	Binary Input (BACnet Regular).....	9-16
360	<b>9.4.9</b>	Binary Input (BACnet Extended).....	9-17
361	<b>9.4.10</b>	Binary Output (BACnet Regular) .....	9-19
362	<b>9.4.11</b>	Binary Output (BACnet Extended).....	9-20
363	<b>9.4.12</b>	Binary Value (BACnet Regular).....	9-21
364	<b>9.4.13</b>	Binary Value (BACnet Extended).....	9-23
365	<b>9.4.14</b>	Multistate Input (BACnet Regular).....	9-24
366	<b>9.4.15</b>	Multistate Input (BACnet Extended).....	9-25
367	<b>9.4.16</b>	Multistate Output (BACnet Regular) .....	9-26
368	<b>9.4.17</b>	Multistate Output (BACnet Extended).....	9-28
369	<b>9.4.18</b>	Multistate Value (BACnet Regular).....	9-29
370	<b>9.4.19</b>	Multistate Value (BACnet Extended).....	9-30

371	<b>9.4.20</b>	Attributes of BACnet Regular Clusters .....	9-31
372	<b>9.4.21</b>	Attributes of BACnet Extended Clusters .....	9-33
373	9.5	ISO 7818 Protocol Tunnel .....	9-35
374	<b>9.5.1</b>	Scope and Purpose .....	9-35
375	<b>9.5.2</b>	Definitions .....	9-35
376	<b>9.5.3</b>	General Description .....	9-36
377	<b>9.5.4</b>	Overview .....	9-36
378	<b>9.5.5</b>	Server .....	9-36
379	<b>9.5.6</b>	Client .....	9-39
380	9.6	Partition .....	9-39
381	<b>9.6.1</b>	Scope and Purpose .....	9-39
382	<b>9.6.2</b>	Introduction .....	9-39
383	<b>9.6.3</b>	Server .....	9-42
384	<b>9.6.4</b>	Client .....	9-48
385	<b>9.6.5</b>	General Use of Partition Cluster .....	9-48
386	9.7	11073 Protocol Tunnel .....	9-49
387	<b>9.7.1</b>	Overview .....	9-49
388	<b>9.7.2</b>	Server .....	9-50
389	<b>9.7.3</b>	Client .....	9-57
390	Chapter 10	Smart Energy .....	10-1
391	10.1	General Description .....	10-1
392	<b>10.1.1</b>	Introduction .....	10-1
393	<b>10.1.2</b>	Cluster List .....	10-1
394	10.2	Price .....	10-1
395	<b>10.2.1</b>	Overview .....	10-1
396	<b>10.2.2</b>	Server .....	10-2
397	<b>10.2.3</b>	Client .....	10-22
398	<b>10.2.4</b>	Application Guidelines .....	10-24
399	10.3	Demand Response and Load Control .....	10-26
400	<b>10.3.1</b>	Overview .....	10-26
401	<b>10.3.2</b>	Server .....	10-26
402	<b>10.3.3</b>	Client .....	10-33
403	<b>10.3.4</b>	Application Guidelines .....	10-38
404	<b>10.3.5</b>	Rules and Guidelines for Overlapping Events .....	10-42
405	10.4	Metering .....	10-49
406	<b>10.4.1</b>	Overview .....	10-49
407	<b>10.4.2</b>	Server .....	10-53
408	<b>10.4.3</b>	Metering Application Guidelines .....	10-87
409	10.5	Messaging .....	10-89
410	<b>10.5.1</b>	Overview .....	10-89
411	<b>10.5.2</b>	Server .....	10-89
412	<b>10.5.3</b>	Client .....	10-92
413	<b>10.5.4</b>	Application Guidelines .....	10-94
414	10.6	Tunneling .....	10-94
415	<b>10.6.1</b>	Overview .....	10-95
416	<b>10.6.2</b>	Server .....	10-99
417	<b>10.6.3</b>	Client .....	10-109
418	10.7	Key Establishment .....	10-109
419	<b>10.7.1</b>	Scope and Purpose .....	10-109
420	<b>10.7.2</b>	General Description .....	10-109
421	<b>10.7.3</b>	Cluster List .....	10-112
422	<b>10.7.4</b>	Application Implementation .....	10-123
423	<b>10.7.5</b>	Key Establishment Test Vectors .....	10-129
424	Chapter 11	Over-the-Air Upgrade .....	11-1

425	11.1 Introduction .....	11-1
426	<b>11.1.1</b> Purpose.....	11-1
427	<b>11.1.2</b> Scope.....	11-1
428	<b>11.1.3</b> Terminology .....	11-1
429	11.2 General Description .....	11-1
430	<b>11.2.1</b> Introduction .....	11-1
431	<b>11.2.2</b> Cluster List.....	11-2
432	11.3 OTA Upgrade.....	11-3
433	<b>11.3.1</b> Overview .....	11-3
434	<b>11.3.2</b> Security .....	11-4
435	<b>11.3.3</b> Image Verification .....	11-4
436	<b>11.3.4</b> Image Transport.....	11-5
437	<b>11.3.5</b> Image Signature .....	11-5
438	<b>11.3.6</b> Image Integrity Code.....	11-5
439	11.4 OTA File Format .....	11-6
440	<b>11.4.1</b> General Structure .....	11-6
441	<b>11.4.2</b> OTA Header Format .....	11-6
442	<b>11.4.3</b> Sub-element Format .....	11-11
443	<b>11.4.4</b> Tag Identifiers.....	11-11
444	<b>11.4.5</b> Crypto Suites .....	11-12
445	<b>11.4.6</b> ECDSA Signature Sub-element (Crypto Suite 1) .....	11-12
446	<b>11.4.7</b> ECDSA Signing Certificate Sub-element.....	11-12
447	<b>11.4.8</b> Image Integrity Code Sub-element.....	11-13
448	<b>11.4.9</b> ECDSA Signature Sub-element (Crypto Suite 2) .....	11-13
449	<b>11.4.10</b> ECDSA Signing Certificate Sub-element (Crypto Suite 2) .....	11-14
450	11.5 OTA File Naming .....	11-14
451	11.6 Signatures .....	11-14
452	11.7 ECDSA Signature Calculation.....	11-14
453	<b>11.7.1</b> ECDSA Signature Verification .....	11-15
454	<b>11.7.2</b> Image Integrity Code.....	11-16
455	11.8 Discovery of the Upgrade Server .....	11-17
456	<b>11.8.1</b> Server and Client .....	11-18
457	<b>11.8.2</b> Sleepy Devices.....	11-18
458	11.9 Dependencies.....	11-18
459	11.10 OTA Cluster Attributes .....	11-19
460	<b>11.10.1</b> UpgradeServerID Attribute .....	11-20
461	<b>11.10.2</b> FileOffset Attribute .....	11-20
462	<b>11.10.3</b> CurrentFileVersion Attribute.....	11-20
463	<b>11.10.4</b> CurrentZigBeeStackVersion Attribute .....	11-20
464	<b>11.10.5</b> DownloadedFileVersion Attribute .....	11-20
465	<b>11.10.6</b> DownloadedZigBeeStackVersion Attribute .....	11-20
466	<b>11.10.7</b> ImageUpgradeStatus Attribute.....	11-20
467	<b>11.10.8</b> Manufacturer ID Attribute.....	11-21
468	<b>11.10.9</b> Image Type ID Attribute .....	11-21
469	<b>11.10.10</b> ..... MinimumBlockPeriod Attribute	
470	11-21	
471	<b>11.10.11</b> ..... Image Stamp Attribute	
472	11-22	
473	<b>11.10.12</b> ..... UpgradeActivationPolicy Attribute	
474	11-22	
475	<b>11.10.13</b> ..... UpgradeTimeoutPolicy Attribute	
476	11-22	
477	11.11 OTA Cluster Parameters .....	11-23
478	<b>11.11.1</b> QueryJitter Parameter.....	11-24
479	<b>11.11.2</b> DataSize Parameter .....	11-24
480	<b>11.11.3</b> OTAImageData Parameter .....	11-24

481	<b>11.11.4</b>	CurrentTime and UpgradeTime/RequestTime Parameters .....	11-24
482	11.12	OTA Upgrade Diagram .....	11-26
483	11.13	Command Frames .....	11-27
484	<b>11.13.1</b>	OTA Cluster Command Identifiers .....	11-27
485	<b>11.13.2</b>	OTA Cluster Status Codes .....	11-28
486	<b>11.13.3</b>	Image Notify Command .....	11-28
487	<b>11.13.4</b>	Query Next Image Request Command .....	11-31
488	<b>11.13.5</b>	Query Next Image Response Command .....	11-33
489	<b>11.13.6</b>	Image Block Request Command .....	11-35
490	<b>11.13.7</b>	Image Page Request Command .....	11-37
491	<b>11.13.8</b>	Image Block Response Command .....	11-40
492	<b>11.13.9</b>	Upgrade End Request Command .....	11-44
493	<b>11.13.10</b>	..... Query Device Specific File Request Command	
494		11-48	
495	<b>11.13.11</b>	..... Query Device Specific File Response Command	
496		11-49	
497	11.14	Multiple Files Required for a Bootload .....	11-51
498	<b>11.14.1</b>	Single OTA File with multiple sub-elements .....	11-51
499	<b>11.14.2</b>	Separate OTA Files Upgraded Independently .....	11-51
500	<b>11.14.3</b>	Multiple OTA Files Dependent on Each Other .....	11-52
501	11.15	OTA Upgrade Cluster Management .....	11-52
502	<b>11.15.1</b>	Query Upgrade Status .....	11-52
503	<b>11.15.2</b>	Query Downloaded ZigBee Stack and File Versions .....	11-52
504	<b>11.15.3</b>	Rate Limiting .....	11-53
505	<b>11.15.4</b>	Current Time, Request Time, and MinimumBlockPeriod .....	11-54
506	11.16	OTA Upgrade Process .....	11-55
507	11.17	Application Standard Specific Decisions .....	11-55
508	<b>11.17.1</b>	SE Profile Standard: OTA Upgrade from SE 1.x to SE 2.0 .....	11-56
509	11.18	OTA Upgrade Recovery .....	11-56
510	Chapter 12	Telecommunication .....	12-1
511	12.1	General Description .....	12-1
512	<b>12.1.1</b>	Introduction .....	12-1
513	<b>12.1.2</b>	Cluster List .....	12-1
514	12.2	Information .....	12-1
515	<b>12.2.1</b>	Scope and Purpose .....	12-1
516	<b>12.2.2</b>	Cluster List .....	12-2
517	<b>12.2.3</b>	Overview .....	12-4
518	<b>12.2.4</b>	Server .....	12-4
519	<b>12.2.5</b>	Client .....	12-21
520	<b>12.2.6</b>	Payload Formats for Contents Data .....	12-21
521	12.3	Chatting .....	12-25
522	<b>12.3.1</b>	Introduction .....	12-25
523	<b>12.3.2</b>	Server .....	12-26
524	<b>12.3.3</b>	Client .....	12-36
525	12.4	Voice Over ZigBee .....	12-37
526	<b>12.4.1</b>	Scope and Purpose .....	12-37
527	<b>12.4.2</b>	Overview .....	12-37
528	<b>12.4.3</b>	Server .....	12-38
529	<b>12.4.4</b>	Client .....	12-45
530	Chapter 13	Commissioning .....	13-1
531	13.1	General Description .....	13-1
532	<b>13.1.1</b>	13.1.1 Introduction .....	13-1
533	<b>13.1.2</b>	13.1.2 Cluster List .....	13-1
534	13.2	Commissioning .....	13-1



535	<b>13.2.1</b>	Overview .....	13-1
536	<b>13.2.2</b>	Server .....	13-2
537	<b>13.2.3</b>	Client.....	13-16
538	<b>13.2.4</b>	Commissioning EUI-64s .....	13-17
539	13.3	Touchlink Commissioning .....	13-17
540	<b>13.3.1</b>	Overview .....	13-18
541	<b>13.3.2</b>	Server .....	13-19
542	<b>13.3.3</b>	Client.....	13-44
543	<b>13.3.4</b>	Functional Description .....	13-45
544	Chapter 14	Retail.....	14-1
545	14.1	General Description .....	14-1
546	<b>14.1.1</b>	Introduction .....	14-1
547	<b>14.1.2</b>	Cluster List.....	14-1
548	14.2	Retail Tunnel (MSP Tunnel) .....	14-1
549	<b>14.2.1</b>	Overview .....	14-1
550	<b>14.2.2</b>	Server .....	14-2
551	<b>14.2.3</b>	Client.....	14-4
552	14.3	Mobile Device Configuration .....	14-4
553	<b>14.3.1</b>	Overview .....	14-4
554	<b>14.3.2</b>	Server .....	14-5
555	<b>14.3.3</b>	Client.....	14-7
556	14.4	Neighbor Cleaning.....	14-7
557	<b>14.4.1</b>	Overview .....	14-7
558	<b>14.4.2</b>	Server .....	14-8
559	<b>14.4.3</b>	Client.....	14-9
560	14.5	Nearest Gateway.....	14-10
561	<b>14.5.1</b>	Overview .....	14-10
562	<b>14.5.2</b>	Server .....	14-10
563	<b>14.5.3</b>	Client.....	14-11
564	<b>14.5.4</b>	Examples of Use .....	14-11
565	Chapter 15	Appliance.....	15-1
566	15.1	General Description .....	15-1
567	<b>15.1.1</b>	Introduction .....	15-1
568	<b>15.1.2</b>	Cluster List.....	15-1
569	15.2	EN50523 Appliance Control .....	15-1
570	<b>15.2.1</b>	Overview .....	15-1
571	<b>15.2.2</b>	General Description .....	15-2
572	<b>15.2.3</b>	Server Attributes.....	15-2
573	<b>15.2.4</b>	Server Commands Received.....	15-4
574	<b>15.2.5</b>	Server Commands Generated .....	15-8
575	<b>15.2.6</b>	Client.....	15-10
576	15.3	EN50523 Appliance Identification.....	15-11
577	<b>15.3.1</b>	Overview .....	15-11
578	<b>15.3.2</b>	Server .....	15-11
579	<b>15.3.3</b>	Client.....	15-15
580	15.4	EN50523 Appliance Events and Alerts .....	15-15
581	<b>15.4.1</b>	Overview .....	15-15
582	<b>15.4.2</b>	Server .....	15-17
583	<b>15.4.3</b>	Client.....	15-21
584	15.5	Appliance Statistics.....	15-21
585	<b>15.5.1</b>	Overview .....	15-21
586	<b>15.5.2</b>	Server .....	15-22
587	<b>15.5.3</b>	Client.....	15-25
588	<b>15.5.4</b>	Appliance Statistics Cluster Sequence Diagram .....	15-26



## LIST OF FIGURES

591	Figure 2-1. The ZCL Client Server Model.....	2-3
592	Figure 2-2. Format of the General ZCL Frame .....	2-7
593	Figure 2-3. Format of the Frame Control Field .....	2-7
594	Figure 2-4. Values of the Frame Type Sub-field.....	2-8
595	Figure 2-5. Format of the Read Attributes Command Frame.....	2-10
596	Figure 2-6. Format of Read Attributes Response Command Frame.....	2-11
597	Figure 2-7. Format of the Read Attributes Status Record Field .....	2-11
598	Figure 2-8. Format of the Attribute Value Field for an Array, Set or Bag .....	2-12
599	Figure 2-9. Format of the Attribute Value Field for a Structure.....	2-12
600	Figure 2-10. Format of the Write Attributes Command Frame .....	2-13
601	Figure 2-11. Format of the Write Attribute Record Field .....	2-13
602	Figure 2-12. Format of Write Attributes Response Command Frame .....	2-15
603	Figure 2-13. Format of the Write Attribute Status Record Field.....	2-15
604	Figure 2-14. Write Attributes No Response Command Frame .....	2-16
605	Figure 2-15. Format of the Configure Reporting Command Frame .....	2-17
606	Figure 2-16. Format of the Attribute Reporting Configuration Record .....	2-17
607	Figure 2-17. Format of the Configure Reporting Response Command Frame.....	2-20
608	Figure 2-18. Format of the Attribute Status Record Field.....	2-20
609	Figure 2-19. Read Reporting Configuration Command Frame .....	2-21
610	Figure 2-20. Format of the Attribute Status Record Field.....	2-22
611	Figure 2-21. Format of the Read Reporting Configuration Response Command Frame .....	2-22
612	Figure 2-22. Attribute Reporting Configuration Record Field.....	2-23
613	Figure 2-23. Format of the Report Attributes Command Frame .....	2-24
614	Figure 2-24. Format of the Attribute Report Fields.....	2-25
615	Figure 2-25. Format of the Default Response Command Frame.....	2-27
616	Figure 2-26. Format of the Discover Attributes Command Frame.....	2-28
617	Figure 2-27. Discover Attributes Response Command Frame .....	2-29
618	Figure 2-28. Format of the Attribute Report Fields.....	2-29
619	Figure 2-29. Format of Read Attributes Structured Command Frame.....	2-30
620	Figure 2-30. Format of the Selector Field.....	2-30
621	Figure 2-31. Write Attributes Structured Command Frame.....	2-31
622	Figure 2-32. Format of the Write Attribute Record Field .....	2-32
623	Figure 2-33. Format of the Selector Field.....	2-32
624	Figure 2-34. Write Attributes Structured Response Command Frame .....	2-34
625	Figure 2-35. Format of the Write Attribute Status Record Field .....	2-34
626	Figure 2-36. Format of the Discover Server Commands Command Frame .....	2-36
627	Figure 2-37. Format of the Discover Commands Received Response Frame .....	2-37
628	Figure 2-38. Format of the Discover Attributes Extended Command Frame.....	2-38
629	Figure 2-39. Format of the Discover Attributes Extended Response Command Frame .....	2-39
630	Figure 2-40. Format of the Extended Attribute Information Fields.....	2-40
631	Figure 2-41. Format of the Attribute Access Control Field .....	2-40
632	Figure 2-42. Format of the Semi-precision Number.....	2-47
633	Figure 2-43. Format of the Octet String Type .....	2-48
634	Figure 2-44. Format of the Character String Type .....	2-48
635	Figure 2-45. Format of the Long Octet String Type .....	2-49
636	Figure 2-46. Format of the Long Character String Type .....	2-49
637	Figure 2-47. Format of the Time of Day Type .....	2-51
638	Figure 2-48. Format of the Date Type .....	2-51
639	Figure 3-1. Typical Usage of Device Configuration and Installation Clusters.....	3-2
640	Figure 3-2. Typical Usage of On/Off and Level Control Clusters.....	3-3
641	Figure 3-3. Typical Usage of the Alarms Cluster.....	3-3
642	Figure 3-4. Typical Usage of the Location Cluster with Centralized Device.....	3-4
643	Figure 3-5. Example Usage of the Input, Output and Value Clusters.....	3-5
644	Figure 3-6. Format of the ProductCode attribute.....	3-10

645	Figure 3-7. Format of Identify Query Response Command Payload.....	3-31
646	Figure 3-8. Format of the Trigger Effect Command.....	3-32
647	Figure 3-9. Format of Identify Query Response Command Payload.....	3-33
648	Figure 3-10. Format of the Add Group Command Payload.....	3-36
649	Figure 3-11. Format of the View Group Command Payload.....	3-37
650	Figure 3-12. Format of Get Group Membership Command Payload.....	3-37
651	Figure 3-13. Format of the Remove Group Command Payload.....	3-38
652	Figure 3-14. Add Group If Identifying Command Payload.....	3-39
653	Figure 3-15. Format of the Add Group Response Command Payload.....	3-40
654	Figure 3-16. Format of the View Group Response Command Payload.....	3-40
655	Figure 3-17. Format of the Get Group Membership Response Command Payload.....	3-41
656	Figure 3-18. Format of Remove Group Response Command Payload.....	3-41
657	Figure 3-19. Format of the Add Scene Command Payload.....	3-46
658	Figure 3-20. Format of the View Scene Command Payload.....	3-46
659	Figure 3-21. Format of the Remove Scene Command Payload.....	3-47
660	Figure 3-22. Format of the Remove All Scenes Command Payload.....	3-47
661	Figure 3-23. Format of the Store Scene Command Payload.....	3-48
662	Figure 3-24. Format of the Recall Scene Command Payload.....	3-48
663	Figure 3-25. Format of Get Scene Membership Command Payload.....	3-49
664	Figure 3-26. Format of the Copy Scene Command.....	3-50
665	Figure 3-27. Format of the Mode Field of the Copy Scene Command.....	3-50
666	Figure 3-28. Format of the Add Scene Response Command Payload.....	3-52
667	Figure 3-29. Format of the View Scene Response Command Payload.....	3-52
668	Figure 3-30. Format of Remove Scene Response Command Payload.....	3-53
669	Figure 3-31. Format of the Remove All Scenes Response Command Payload.....	3-53
670	Figure 3-32. Format of the Store Scene Response Command Payload.....	3-53
671	Figure 3-33. Format of the Get Scene Membership Response CommandPayload.....	3-54
672	Figure 3-34. Format of the Copy Scene Response Command.....	3-55
673	Figure 3-35. State Behavior of Store and Recall Global Scene.....	3-58
674	Figure 3-36. Format of the Off With Effect Command.....	3-60
675	Figure 3-37. Format of the On With Timed Off Command.....	3-61
676	Figure 3-38. Format of the On/Off Control Field of the On With Timed Off Command.....	3-62
677	Figure 3-39. On/Off Cluster Operation State Machine.....	3-63
678	Figure 3-40. Format of the Move to Level Command Payload.....	3-72
679	Figure 3-41. Format of the Move Command Payload.....	3-73
680	Figure 3-42. Format of the Step Command Payload.....	3-74
681	Figure 3-43. Format of the Command Payload.....	3-75
682	Figure 3-44. Format of the Command Payload.....	3-75
683	Figure 3-45. Format of the Reset Alarm Command Payload.....	3-79
684	Figure 3-46. Format of the Alarm Command Payload.....	3-80
685	Figure 3-47. Format of the Get Alarm Response Command Payload.....	3-80
686	Figure 3-48. Example of Usage of RSSI Location Cluster.....	3-85
687	Figure 3-49. Format of the Set Absolute Location Command Payload.....	3-90
688	Figure 3-50. Format of the Set Device Configuration Payload.....	3-90
689	Figure 3-51. Format of the Get Device Configuration Payload.....	3-91
690	Figure 3-52. Format of the Get Location Data Payload.....	3-91
691	Figure 3-53. Format of the RSSI Response Command Payload.....	3-93
692	Figure 3-54. Format of the Send Pings Command Payload.....	3-93
693	Figure 3-55. Format of the Anchor Node Announce Command Payload.....	3-94
694	Figure 3-56. Format of the Device Configuration Response Payload.....	3-95
695	Figure 3-57. Format of the Location Data Response Payload.....	3-96
696	Figure 3-58. Format of the Location Data Notification Payload.....	3-96
697	Figure 3-59. Format of the RSSI Ping Command Payload.....	3-97
698	Figure 3-60. Format of the Report RSSI Measurements Command Payload.....	3-98
699	Figure 3-61. Neighbor Info Structure.....	3-98
700	Figure 3-62. Format of the Request Own Location Command Payload.....	3-99

701	Figure 3-63. Format of the Check-in Response Payload.....	3-159
702	Figure 3-64. Format of the Set Long Poll Interval Command Payload.....	3-160
703	Figure 3-65. Format of the Set Short Poll Interval Command Payload .....	3-161
704	Figure 3-66. Poll Control Cluster Sequence Diagram .....	3-162
705	Figure 3-67. Typical Usage of the Power Profile Cluster .....	3-164
706	Figure 3-68. Format of the PowerProfileRequest Command Payload .....	3-167
707	Figure 3-69. Format of the GetPowerProfilePriceResponse Command.....	3-168
708	Figure 3-70. Format of the GetOverallSchedulePriceResponse Command .....	3-169
709	Figure 3-71. Format of the EnergyPhasesScheduleNotification Command Payload.....	3-170
710	Figure 3-72. Format of the PowerProfileNotification Command Payload (1 of 2).....	3-175
711	Figure 3-73. Format of the PowerProfileStateResponse Command Frame.....	3-177
712	Figure 3-74. Format of the Power Profile Record Field .....	3-177
713	Figure 3-75. Power Profile States.....	3-179
714	Figure 3-76. Power Profile State Diagram.....	3-179
715	Figure 3-77. Format of EnergyPhasesScheduleStateResponse in Case of No Scheduled Phases.....	3-182
716	Figure 3-78. Format of the PowerProfileScheduleConstraintsNotification Command Frame .....	3-183
717	Figure 3-79. Format of the GetPowerProfilePriceExtended Command Payload.....	3-184
718	Figure 3-80. Visualization of Price Associated to a Power Profile.....	3-186
719	Figure 3-81. Energy Remote Disabled: Example of Sequence Diagram with User Interaction .....	3-187
720	Figure 3-82. Energy Remote Enabled: Example of Sequence Diagram with User Interaction .....	3-188
721	Figure 3-83. Typical Usage of the Meter Identification Cluster.....	3-189
722	Figure 4-1. Typical Usage of Illuminance Measurement and Level Sensing Clusters.....	4-2
723	Figure 4-2. Typical Usage of Temperature, Pressure and Flow Measurement Clusters .....	4-3
724	Figure 4-3. Typical Usage of Occupancy Sensing Cluster.....	4-4
725	Figure 4-4. The DC Overload Alarm Mask .....	4-35
726	Figure 4-5. The ACAlarmsMask Attribute .....	4-36
727	Figure 4-6. Format of the Get Profile Info Response Command.....	4-43
728	Figure 4-7. ProfileIntervalPeriod.....	4-43
729	Figure 4-8. Format of the Get Measurement Profile Response Command.....	4-44
730	Figure 4-9. Format of the Get Measurement Profile Command.....	4-45
731	Figure 5-1. Typical Usage of Ballast Configuration and Color Control Clusters.....	5-2
732	Figure 5-2. Format of the Move to Hue Command Payload .....	5-16
733	Figure 5-3. Format of the Move Hue Command Payload .....	5-17
734	Figure 5-4. Format of the Step Hue Command Payload .....	5-18
735	Figure 5-5. Format of the Move to Saturation Command Payload.....	5-19
736	Figure 5-6. Format of the Move Saturation Command Payload.....	5-20
737	Figure 5-7. Format of the Step Saturation Command Payload.....	5-21
738	Figure 5-8. Move to Hue and Saturation Command Payload.....	5-22
739	Figure 5-9. Format of the Move to Color Command Payload.....	5-23
740	Figure 5-10. Format of the Move Color Command Payload.....	5-23
741	Figure 5-11. Format of the Step Color Command Payload.....	5-24
742	Figure 5-12. Move to Color Temperature Command Payload .....	5-25
743	Figure 5-13. Format of the Enhanced Move to Hue Command .....	5-25
744	Figure 5-14. Format of the Enhanced Move Hue Command .....	5-26
745	Figure 5-15. Format of the Enhanced Step Hue Command .....	5-27
746	Figure 5-16. Format of the Enhanced Move to Hue and Saturation Command.....	5-28
747	Figure 5-17. Format of the Color Loop Set Command.....	5-29
748	Figure 5-18. Format of the Update Flags Field of the Color Loop Set Command.....	5-29
749	Figure 5-19. Format of the Stop Move Step Command Payload.....	5-31
750	Figure 5-20. Format of the Move Color Temperature Command.....	5-31
751	Figure 5-21. Format of the Step Color Temperature Command.....	5-33
752	Figure 6-1. Typical Usage of Pump Configuration and Control Cluster.....	6-2
753	Figure 6-2. Example Usage of the Thermostat and Related Clusters .....	6-2
754	Figure 6-3. Priority Scheme of Pump Operation and Control .....	6-10
755	Figure 6-4. Format of the Setpoint Raise/Lower Command Payload .....	6-30
756	Figure 6-5. Set Weekly Schedule Command Payload Format (1 of 2).....	6-30

757	Figure 6-6. Set Weekly Schedule Command Payload Format (2 of 2).....	6-30
758	Figure 6-7. Set Heat Weekly Schedule Command Payload Format (1 of 2).....	6-32
759	Figure 6-8. Set Heat Weekly Schedule Command Payload Format (2 of 2).....	6-32
760	Figure 6-9. Set Cool Weekly Schedule Command Payload Format (1 of 2).....	6-32
761	Figure 6-10. Set Cool Weekly Schedule Command Payload Format (2 of 2).....	6-32
762	Figure 6-11. Set Heat & Cool Weekly Schedule Command Payload Format (1 of 2).....	6-32
763	Figure 6-12. Set Heat & Cool Weekly Schedule Command Payload Format (2 of 2).....	6-33
764	Figure 6-13. Format of the Get Weekly Schedule Command Payload.....	6-34
765	Figure 6-14. Format of the Relay Status Log Payload.....	6-36
766	Figure 7-1. Typical Usage of the Closures Clusters.....	7-2
767	Figure 7-2. Format of the Alarm Cluster.....	7-6
768	Figure 7-3. Format of the Lock Door Command.....	7-20
769	Figure 7-4. Format of the Unlock Door Command.....	7-20
770	Figure 7-5. Format of the Toggle Command.....	7-20
771	Figure 7-6. Format of the Unlock with Timeout Command.....	7-21
772	Figure 7-7. Format of the Get Log Record Command.....	7-21
773	Figure 7-8. Format of the Set PIN Code Command.....	7-22
774	Figure 7-9. Format of the Get PIN Code Command.....	7-23
775	Figure 7-10. Format of the Clear PIN Code Command.....	7-23
776	Figure 7-11. Format of the Set User Status Command.....	7-23
777	Figure 7-12. Format of the Get User Status Command.....	7-24
778	Figure 7-13. Format of the Set Week Day Schedule Command.....	7-24
779	Figure 7-14. Format of Days Mask Bits.....	7-24
780	Figure 7-15. Format of the Get Week Day Schedule Command.....	7-25
781	Figure 7-16. Format of the Clear Week Day Schedule Command.....	7-25
782	Figure 7-17. Format of the Set Year Day Schedule Command.....	7-25
783	Figure 7-18. Format of the Get Year Day Schedule Command.....	7-26
784	Figure 7-19. Format of the Clear Year Day Schedule Command.....	7-26
785	Figure 7-20. Format of the Set Holiday Schedule Command.....	7-26
786	Figure 7-21. Format of the Get Holiday Schedule Command.....	7-27
787	Figure 7-22. Format of the Clear Holiday Schedule Command.....	7-27
788	Figure 7-23. Format of the Set User Type Command.....	7-27
789	Figure 7-24. Format of the Get User Type Command.....	7-27
790	Figure 7-25. Format of the Set RFID Code Command.....	7-28
791	Figure 7-26. Format of the Get RFID Code Command.....	7-28
792	Figure 7-27. Format of the Clear RFID Code Command.....	7-29
793	Figure 7-28. Format of the Lock Door Response Command Payload.....	7-31
794	Figure 7-29. Format of the Unlock Door Response Command Payload.....	7-31
795	Figure 7-30. Format of the Get Log Record Response Command.....	7-32
796	Figure 7-31. Format of the Set PIN Code Response Command.....	7-33
797	Figure 7-32. Format of the Get PIN Code Response Command.....	7-33
798	Figure 7-33. Format of the Clear PIN Code Response Command.....	7-33
799	Figure 7-34. Format of the Clear All PIN Codes Response Command.....	7-34
800	Figure 7-35. Format of the Set User Status Response Command.....	7-34
801	Figure 7-36. Format of the Get User Status Response Command.....	7-34
802	Figure 7-37. Format of the Set Week Day Schedule Response Command.....	7-35
803	Figure 7-38. Format of the Get Week Day Schedule Response Command.....	7-35
804	Figure 7-39. Format of Days Mask Bits.....	7-35
805	Figure 7-40. Format of the Clear Week Day Schedule ID Response Command.....	7-36
806	Figure 7-41. Format of the Set Year Day Schedule Response Command.....	7-36
807	Figure 7-42. Format of the Get Year Day Schedule Response Command.....	7-37
808	Figure 7-43. Format of the Clear Year Day Schedule Response Command.....	7-37
809	Figure 7-44. Format of the Set Holiday Schedule Response Command.....	7-38
810	Figure 7-45. Format of the Get Holiday Schedule Response Command.....	7-38
811	Figure 7-46. Format of the Clear Holiday Schedule Response Command.....	7-39
812	Figure 7-47. Format of the Set User Type Response Command.....	7-39

813	Figure 7-48. Format of the Get User Type Response Command.....	7-39
814	Figure 7-49. Format of the Set RFID Code Response Command .....	7-40
815	Figure 7-50. Format of the Get RFID Code Response Command.....	7-40
816	Figure 7-51. Format of the Clear RFID Code Response Command .....	7-40
817	Figure 7-52. Format of the Clear All RFID Codes Response Command.....	7-41
818	Figure 7-53. Format of the Operation Event Notification Command .....	7-41
819	Figure 7-54. Format of the Programming Event Notification Command .....	7-45
820	Figure 7-55. Format of the Go To Lift Value Command.....	7-56
821	Figure 7-56. Format of the Go To Lift Percentage Command .....	7-56
822	Figure 7-57. Format of the Go To Tilt Value Command .....	7-57
823	Figure 7-58. Format of the Go To Lift Percentage Command .....	7-57
824	Figure 8-1. Typical Usage of the IAS Clusters .....	8-2
825	Figure 8-2. Format of the Zone Enroll Response Command Payload .....	8-9
826	Figure 8-3. Payload format of Initiate Test Mode command .....	8-10
827	Figure 8-4. Format of the Zone Status Change Notification Command Payload .....	8-12
828	Figure 8-5. Format of the Zone Enroll Request Command Payload.....	8-12
829	Figure 8-6. Format of the Arm Command Payload .....	8-15
830	Figure 8-7. Format of the Bypass Command Payload .....	8-16
831	Figure 8-8. Format of the Get Zone Information Command Payload.....	8-17
832	Figure 8-9. Format of the Get Zone Status command.....	8-18
833	Figure 8-10. Format of the Arm Response Command Payload.....	8-19
834	Figure 8-11. Get Zone ID Map Response Command Payload .....	8-20
835	Figure 8-12. Format of the Get Zone Information Response Command Payload .....	8-20
836	Figure 8-13. Format of the Zone Status Changed Command Payload.....	8-21
837	Figure 8-14. Audible Notification field value .....	8-21
838	Figure 8-15. Format of the Panel Status Changed Command Payload .....	8-22
839	Figure 8-16. Alarm Status field value.....	8-23
840	Figure 8-17. Get Panel Status Response command .....	8-24
841	Figure 8-18. Set Bypassed Zone List Command payload format .....	8-24
842	Figure 8-19. Bypass Response command format .....	8-25
843	Figure 8-20. Format of the Get Zone Status Response command .....	8-26
844	Figure 8-21. Format of the Start Siren Command Payload.....	8-29
845	Figure 8-22. Format of the Start Siren Command Payload.....	8-31
846	Figure 9-1. Format of Match Protocol Address Command Payload.....	9-4
847	Figure 9-2. Match Protocol Address Response Command Payload .....	9-5
848	Figure 9-3. Advertise Protocol Address Command Payload.....	9-5
849	Figure 9-4. Format of the Transfer NPDU Command Payload.....	9-7
850	Figure 9-5. Format of the Transfer APDU command.....	9-37
851	Figure 9-6. Typical Usage of the Partition Cluster .....	9-40
852	Figure 9-7. Client and Server in Partition Cluster .....	9-41
853	Figure 9-8. Format of the TransferPartitionedFrame Command.....	9-44
854	Figure 9-9. Format of the FragmentationOptions Field .....	9-45
855	Figure 9-10. ReadHandshakeParam Frame.....	9-45
856	Figure 9-11. WriteHandshakeParam Frame.....	9-46
857	Figure 9-12. Format of Write Attribute Record Field.....	9-46
858	Figure 9-13. Format of the MultipleACK Command .....	9-46
859	Figure 9-14. Format of the ACK Options Field .....	9-46
860	Figure 9-15. ReadHandshakeParamResponse Frame .....	9-47
861	Figure 9-16. Format of Read Attribute Status Record Field .....	9-48
862	Figure 9-17. Example of Partition Cluster Use .....	9-49
863	Figure 9-18 Typical Usage of the 11073 Protocol Tunnel cluster .....	9-50
864	Figure 9-19 – Transfer APDU payload.....	9-53
865	Figure 9-20 – Connect Request command payload .....	9-54
866	Figure 9-21 – Connect control field format .....	9-54
867	Figure 9-22 – Disconnect Request command payload.....	9-55
868	Figure 9-23 – Connect Status Notification command payload.....	9-56

869	Figure 10-1. Price Cluster Client Server Example.....	10-2
870	Figure 10-2. The Format of the Get Current Price Command Payload.....	10-11
871	Figure 10-3. Get Current Price Command Options Field .....	10-11
872	Figure 10-4. Format of the Get Scheduled Prices Command Payload.....	10-12
873	Figure 10-5. Format of the Price Acknowledgement Command Payload .....	10-13
874	Figure 10-6. Format of the Get Block Period(s) Command Payload .....	10-13
875	Figure 10-7. Format of the GetConversionFactor Command Payload.....	10-14
876	Figure 10-8. Format of the GetCalorificValue Command Payload .....	10-15
877	Figure 10-9. Format of the Publish Price Command Payload .....	10-16
878	Figure 10-10. Format of the Publish Block Period Command Payload .....	10-20
879	Figure 10-11. Format of the PublishConversionFactor Command Payload.....	10-21
880	Figure 10-12. Format of the PublishCalorificValue Command Payload.....	10-22
881	Figure 10-13. Demand Response/Load Control Cluster Client Server Example.....	10-26
882	Figure 10-14. Format of the Load Control Event Command Payload .....	10-27
883	Figure 10-15. Format of the Cancel Load Control Event Payload.....	10-31
884	Figure 10-16. Format of the Report Event Status Command Payload .....	10-35
885	Figure 10-17. Format of the Get Scheduled Events Command Payload .....	10-37
886	Figure 10-18. Example of Both a Successful and an Overridden Load Curtailment Event.....	10-41
887	Figure 10-19. Example of a Load Curtailment Superseded and Another Cancelled.....	10-42
888	Figure 10-20. Smart Energy Device Class Reference Example.....	10-45
889	Figure 10-21. Correctly Overlapping Events.....	10-45
890	Figure 10-22. Correct Superseding of Events.....	10-46
891	Figure 10-23. Superseded Event for a Subset of Device Classes.....	10-47
892	Figure 10-24. Ending Randomization Between Events .....	10-47
893	Figure 10-25. Start Randomization Between Events .....	10-48
894	Figure 10-26. Acceptable Gaps with Start and Stop Randomization .....	10-49
895	Figure 10-27. Standalone ESI Model with Mains Powered Metering Device.....	10-50
896	Figure 10-28. Standalone ESI Model with Battery Powered Metering Device .....	10-51
897	Figure 10-29. ESI Model with Integrated Metering Device.....	10-52
898	Figure 10-30. Format of the Get Profile Response Command Payload.....	10-82
899	Figure 10-31. Format of the Request Fast Poll Mode Response Command Payload.....	10-84
900	Figure 10-32. Format of the Get Profile Command Payload.....	10-85
901	Figure 10-33. Format of the Request Mirror Response Command Payload.....	10-86
902	Figure 10-34. Format of the Mirror Removed Command Payload .....	10-86
903	Figure 10-35. Format of the Request Fast Poll Mode Command Payload .....	10-87
904	Figure 10-36. Messaging Cluster Client/Server Example.....	10-89
905	Figure 10-37. Format of the Display Message Command Payload.....	10-90
906	Figure 10-38. Format of the Cancel Message Command Payload.....	10-92
907	Figure 10-39. Format of the Message Confirmation Command Payload.....	10-93
908	Figure 10-40. Client/Server Message Command Exchanges .....	10-94
909	Figure 10-41. A Client Requests a Tunnel from a Server to Exchange Complex Data in Both Directions.....	10-95
910	Figure 10-42. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data Without Flow Control (Default) .....	10-97
911	Figure 10-43. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data with Flow Control .....	10-97
912	Figure 10-44. Format of the RequestTunnel Command Payload.....	10-100
913	Figure 10-45. Format of the CloseTunnel Command Payload .....	10-101
914	Figure 10-46. Format of the TransferData Command Payload .....	10-102
915	Figure 10-47. Format of the TransferDataError Command Payload.....	10-102
916	Figure 10-48. Format of the AckTransferData Command Payload .....	10-103
917	Figure 10-49. Format of the ReadyData Command Payload.....	10-104
918	Figure 10-50. Format of the Get Supported Tunnel Protocols Command Payload .....	10-105
919	Figure 10-51. Format of the RequestTunnelResponse Command Payload .....	10-106
920	Figure 10-52. Format of the TransferData Command Payload .....	10-107
921	Figure 10-53. Format of the Supported Tunnel Protocols Response Command Payload.....	10-107
922	Figure 10-54. Format of the Supported Tunnel Protocols Response Command Protocol Fields .....	10-108



925	Figure 10-55. Format of the TunnelClosureNotification Command Payload.....	10-108
926	Figure 10-56. Overview of General Exchange .....	10-111
927	Figure 10-57. Typical Usage of the Key Establishment Cluster.....	10-113
928	Figure 10-58. Key Establishment Command Exchange .....	10-114
929	Figure 10-59. Format of the Initiate Key Establishment Request Command Payload.....	10-116
930	Figure 10-60. Format of the Ephemeral Data Request Command Payload.....	10-117
931	Figure 10-61. Format of the Confirm Key Request Command Payload .....	10-117
932	Figure 10-62. Format of the Terminate Key Establishment Command Payload .....	10-118
933	Figure 10-63. Format of the Initiate Key Establishment Response Command Payload .....	10-120
934	Figure 10-64. Format of the Ephemeral Data Response Command Payload .....	10-121
935	Figure 10-65. Format of the Confirm Key Response Command Payload .....	10-121
936	Figure 10-66. Format of the Terminate Key Establishment Command Payload .....	10-122
937	Figure 10-67. Key Establishment Command Exchange .....	10-132
938	Figure 11-1. Typical Usage of OTA Upgrade Cluster .....	11-2
939	Figure 11-2. Sample OTA File.....	11-6
940	Figure 11-3. Sub-element Format.....	11-11
941	Figure 11-4. ECDSA Signature.....	11-12
942	Figure 11-5. ECDSA Signing Certificate Sub-element .....	11-13
943	Figure 11-6. Hash Value Sub-element.....	11-13
944	Figure 11-7. ECDSA Signature.....	11-13
945	Figure 11-8. ECDSA Signing Certificate Sub-element .....	11-14
946	Figure 11-9. OTA Upgrade Message Diagram .....	11-26
947	Figure 11-10. Format of Image Notify Command Payload.....	11-29
948	Figure 11-11. Format of Query Next Image Request Command Payload.....	11-31
949	Figure 11-12. Format of Query Next Image Response Command Payload .....	11-33
950	Figure 11-13. Format of Image Block Request Command Payload .....	11-35
951	Figure 11-14. Image Page Request Command Payload .....	11-37
952	Figure 11-15. Image Block Response Command Payload with SUCCESS status .....	11-40
953	Figure 11-16. Image Block Response Command Payload with WAIT_FOR_DATA status .....	11-41
954	Figure 11-17. Image Block Response Command Payload with ABORT status.....	11-41
955	Figure 11-18. Format of Upgrade End Request Command Payload.....	11-44
956	Figure 11-19. Format of Upgrade End Response Command Payload .....	11-46
957	Figure 11-20. Format of Query Device Specific File Request Command Payload .....	11-48
958	Figure 11-21. Format of Query Device Specific File Response Command Payload .....	11-49
959	Figure 11-22. Rate Limiting Exchange.....	11-54
960	Figure 12-1. Typical Content Data Structure .....	12-2
961	Figure 12-2. Typical Usage of the Information Cluster .....	12-3
962	Figure 12-3. Typical Usage of the Information Cluster – with Proxy Function .....	12-4
963	Figure 12-4. An Example Sequence .....	12-5
964	Figure 12-5. Preference Scenarios (Triggered by the Client or by the Server).....	12-6
965	Figure 12-6. Payload Format of Request Information Command .....	12-9
966	Figure 12-7. Payload Format for Request a Content by a Content ID .....	12-10
967	Figure 12-8. Request Information Payload for Request Contents by Multiple IDs .....	12-10
968	Figure 12-9. Request Information Payload for Request by Depth.....	12-11
969	Figure 12-10. Payload Format of Push Information Response Command .....	12-11
970	Figure 12-11. Payload Format for Send Preference Command.....	12-12
971	Figure 12-12. Payload Format for Preference Is Multiple Content ID (0x0000).....	12-12
972	Figure 12-13. Payload Format for Preference Is Multiple Octet Strings (0x0001).....	12-12
973	Figure 12-14. Payload Format of Request Preference Response Command.....	12-13
974	Figure 12-15. Payload Format for Update Command.....	12-13
975	Figure 12-16. Format for Redirection Control Field.....	12-14
976	Figure 12-17. An Example Sequence of Forwarding Case .....	12-15
977	Figure 12-18. An Example Sequence of Redirecting Case .....	12-16
978	Figure 12-19. Payload Format for Delete Command.....	12-16
979	Figure 12-20. Format for Deletion Option Field .....	12-17
980	Figure 12-21. Payload Format for Configure Node Description Command .....	12-17

981	Figure 12-22. Payload Format for Configure Delivery Enable Command.....	12-18
982	Figure 12-23. Payload Format for Configure Push Information Timer Command.....	12-18
983	Figure 12-24. Payload Format for Configure Set Root ID Command .....	12-18
984	Figure 12-25. Payload Format of Request Information Response Command .....	12-19
985	Figure 12-26. Payload Format of Push Information Command .....	12-19
986	Figure 12-27. Payload Format for Send Preference Response Command and Request Preference Confirmation	
987	Command.....	12-20
988	Figure 12-28. Payload Format of Update Response and Delete Response command.....	12-21
989	Figure 12-29. Payload Format for Multiple Contents .....	12-22
990	Figure 12-30. Format for Single Content.....	12-22
991	Figure 12-31. Format for Title String .....	12-23
992	Figure 12-32. Format for Long Octet String .....	12-23
993	Figure 12-33. Format for Long Character String.....	12-23
994	Figure 12-34. Format for RSS Feed .....	12-23
995	Figure 12-35. Typical Usage of the Chatting Cluster .....	12-26
996	Figure 12-36. Format of the Join Chat Request Command.....	12-29
997	Figure 12-37. Format of the Leave Chat Request Command.....	12-29
998	Figure 12-38. Format of the Switch Chairman Response Command.....	12-30
999	Figure 12-39. Format of the Start Chat Request Command.....	12-30
1000	Figure 12-40. Format of the ChatMessage Command.....	12-31
1001	Figure 12-41. Format of the Get Node Information Request Command.....	12-31
1002	Figure 12-42. Format of an Item of the Chatting Table .....	12-31
1003	Figure 12-43. Format of the Start Chat Response Command.....	12-32
1004	Figure 12-44. Format of the Join Chat Response Command.....	12-33
1005	Figure 12-45. Format of the User Left Command .....	12-33
1006	Figure 12-46. Format of the User Joined Command .....	12-34
1007	Figure 12-47. Format of the Search Chat Response command .....	12-34
1008	Figure 12-48. Format of the Switch Chairman Request Command.....	12-35
1009	Figure 12-49. Format of the Switch Chairman Confirm Command .....	12-35
1010	Figure 12-50. Format of the NodeInformation Field .....	12-35
1011	Figure 12-51. Format of the Switch Chairman Notification Command.....	12-36
1012	Figure 12-52. Format of the Get Node Information Response Command .....	12-36
1013	Figure 12-53. Typical Usage of the VoZ Cluster .....	12-37
1014	Figure 12-54. Format of the OptionFlags Attribute.....	12-40
1015	Figure 12-55. Format of the Establishment Request Command.....	12-41
1016	Figure 12-56. Format of the Flag .....	12-41
1017	Figure 12-57. Format of the Voice Transmission Command.....	12-42
1018	Figure 12-58. Format of the Voice Transmission Completion Command .....	12-42
1019	Figure 12-59. Format of the Control Response Command .....	12-42
1020	Figure 12-60. Format of the Voice Transmission Response Command.....	12-43
1021	Figure 12-61. Format of the Establishment Response Command .....	12-44
1022	Figure 12-62. Format of the Control Command.....	12-45
1023	Figure 13-1. Format of the Restart Device Command Payload.....	13-12
1024	Figure 13-2. Format of the Options Field .....	13-12
1025	Figure 13-3. Format of Save Startup Parameters Command Payload.....	13-14
1026	Figure 13-4. Restore Startup Parameters Command Payload .....	13-14
1027	Figure 13-5. Format of Reset Startup Parameters Command Payload.....	13-15
1028	Figure 13-6. Format of the Options Field .....	13-15
1029	Figure 13-7. Format of Reset Startup Parameters Command Payload.....	13-16
1030	Figure 13-8. Format of the Scan Request Command Frame .....	13-20
1031	Figure 13-9. Format of the ZigBee Information Field .....	13-21
1032	Figure 13-10. Format of the Scan Request Touchlink Information Field .....	13-21
1033	Figure 13-11. Format of the Device Information Request Command Frame .....	13-22
1034	Figure 13-12. Format of the Identify Request Command Frame.....	13-23
1035	Figure 13-13. Format of the Reset to Factory New Request Command Frame.....	13-24
1036	Figure 13-14. Format of the Network Start Request Command Frame .....	13-24

1037	Figure 13-15. Format of the Network Join Router Request Command Frame .....	13-26
1038	Figure 13-16. Format of the Network Join End Device Request Command Frame .....	13-28
1039	Figure 13-17. Format of the Network Update Request Command Frame.....	13-30
1040	Figure 13-18. Format of the Get Group Identifiers Request Command.....	13-31
1041	Figure 13-19. Format of the Get Endpoint List Request Command .....	13-31
1042	Figure 13-20. Format of the Scan Response Command Frame .....	13-32
1043	Figure 13-21. Format of the ZigBee Information Field .....	13-33
1044	Figure 13-22. Format of the Scan Response Touchlink Information Field .....	13-34
1045	Figure 13-23. Format of the Device Information Response Command Frame.....	13-36
1046	Figure 13-24. Format of the Device Information Record Field.....	13-36
1047	Figure 13-25. Format of the Network Start Response Command Frame.....	13-38
1048	Figure 13-26. Format of the Network Join Router Response Command Frame.....	13-39
1049	Figure 13-27. Format of the Network Join End Device Response Command Frame .....	13-40
1050	Figure 13-28. Format of the Endpoint Information Command .....	13-41
1051	Figure 13-29. Format of the Get Group Identifiers Response Command .....	13-42
1052	Figure 13-30. Format of a Group Information Record Entry .....	13-42
1053	Figure 13-31. Format of the Get Endpoint List Response Command.....	13-43
1054	Figure 13-32. Format of an Endpoint Information Record Entry .....	13-43
1055	Figure 13-33. Format of the device information table .....	13-46
1056	Figure 13-34. General format of an inter-PAN frame.....	13-48
1057	Figure 13-35. Scope of a touchlink commissioning inter-PAN transaction .....	13-49
1058	Figure 13-36. Overview of Touchlink Security.....	13-53
1059	Figure 13-37. Steps Required to Encrypt/Decrypt the Network Key.....	13-56
1060	Figure 14-1. Typical Usage of the Retail Tunnel Cluster .....	14-2
1061	Figure 14-2. Format of the Transfer APDU Command .....	14-3
1062	Figure 14-3. Typical Usage of the Mobile Device Configuration Cluster .....	14-5
1063	Figure 14-4. Format of the Keep Alive Notification Command.....	14-6
1064	Figure 14-5. Typical Usage of the Neighbor Cleaning Cluster .....	14-8
1065	Figure 14-6. Typical Usage of the Nearest Gateway Cluster .....	14-10
1066	Figure 14-7. Sequence Diagram.....	14-12
1067	Figure 15-1. Typical Usage of the Appliance Control Cluster .....	15-2
1068	Figure 15-2. Format of the Execution of a Command Payload .....	15-4
1069	Figure 15-3. Format of the Write Functions Command Frame .....	15-6
1070	Figure 15-4. Format of the Write Functions Record Field.....	15-6
1071	Figure 15-5. Format of the Overload Warning Payload.....	15-7
1072	Figure 15-6. Format of the Signal State Response Command Payload.....	15-8
1073	Figure 15-7. Typical Usage of the Appliance Events and Alerts Cluster.....	15-16
1074	Figure 15-8. Format of the Get Alerts Response Command Payload .....	15-18
1075	Figure 15-9. Format of the Alerts Notification Command Payload.....	15-20
1076	Figure 15-10. Format of the Event Notification Command Payload .....	15-20
1077	Figure 15-11. Format of the Log Notification Payload.....	15-24
1078	Figure 15-12. Format of the Log Queue Response Payload.....	15-25
1079	Figure 15-13. Format of the Log Request Payload.....	15-26
1080	Figure 15-14. Appliance Statistics Cluster Sequence Diagram.....	15-27
1081		

1082

## LIST OF TABLES

1083	Table 1-1. Acronyms and Abbreviations .....	1-1
1084	Table 2-1. Global Attributes .....	2-6
1085	Table 2-2. AttributeReportingStatus Enumerations.....	2-6
1086	Table 2-3. ZCL Command Frames.....	2-9
1087	Table 2-4. Destination of Reporting Based on Direction Field .....	2-18
1088	Table 2-5. Valid Profile Identifier Values.....	2-41
1089	Table 2-6. Valid Device Identifier Values .....	2-41
1090	Table 2-7. Valid Cluster Identifier Values .....	2-42
1091	Table 2-8. Valid ZCL Defined Attribute Identifier Values .....	2-42
1092	Table 2-9. Valid ZCL-Defined Command Identifier Values.....	2-42
1093	Table 2-10. Data Types.....	2-43
1094	Table 2-11. Enumerated Status Values Used in the ZCL.....	2-52
1095	Table 3-1. Device Configuration and Installation Clusters .....	3-1
1096	Table 3-2. Groups and Scenes Clusters .....	3-2
1097	Table 3-3. On/Off and Level Control Clusters .....	3-2
1098	Table 3-4. Alarms Cluster.....	3-3
1099	Table 3-5. Other Clusters.....	3-4
1100	Table 3-6. Generic Clusters .....	3-4
1101	Table 3-7. Attributes of the Basic Cluster.....	3-6
1102	Table 3-8. Values of the PowerSource Attribute.....	3-8
1103	Table 3-9. Values of the GenericDeviceClass attribute .....	3-9
1104	Table 3-10. Values of the GenericDeviceType attribute for the lighting class.....	3-9
1105	Table 3-11. Values of the CodeId field of the ProductCode attribute.....	3-11
1106	Table 3-12. Values of the PhysicalEnvironment Attribute.....	3-12
1107	Table 3-13. Values of the DeviceEnable Attribute.....	3-15
1108	Table 3-14. Values of the AlarmMask Attribute.....	3-16
1109	Table 3-15. Values of the DisableLocalConfig Attribute .....	3-16
1110	Table 3-16. Received Command IDs for the Basic Cluster .....	3-16
1111	Table 3-17. Power Configuration Attribute Sets .....	3-18
1112	Table 3-18. Attributes of the Mains Information Attribute Set .....	3-18
1113	Table 3-19. Attributes of the Mains Settings Attribute Set .....	3-19
1114	Table 3-20. Values of the MainsAlarmMask Attribute.....	3-19
1115	Table 3-21. Attributes of the Battery Information Attribute Set.....	3-20
1116	Table 3-22. Attributes of the Battery Settings Attribute Set .....	3-21
1117	Table 3-23. Values of the BatterySize Attribute .....	3-21
1118	Table 3-24. Values of the BatteryAlarmMask Attribute.....	3-22
1119	Table 3-25. Alarm Code Field Enumerations for Battery Alarms.....	3-23
1120	Table 3-26. BatteryAlarmState Enumerations .....	3-25
1121	Table 3-27. Device Temperature Configuration Attribute Sets.....	3-27
1122	Table 3-28. Device Temperature Information Attribute Set .....	3-27
1123	Table 3-29. Device Temperature Settings Attribute Set .....	3-28
1124	Table 3-30. Values of the DeviceTempAlarmMask Attribute .....	3-28
1125	Table 3-31. Attributes of the Identify Server Cluster .....	3-30
1126	Table 3-32. Received Command IDs for the Identify Cluster .....	3-31
1127	Table 3-33. Values of the Effect Identifier Field of the Trigger Effect Command.....	3-32
1128	Table 3-34. Values of the Effect Variant Field of the Trigger Effect Command.....	3-32
1129	Table 3-35. Generated Command IDs for the Identify Cluster .....	3-33
1130	Table 3-36. Attributes of the Groups Server Cluster .....	3-35
1131	Table 3-37. Received Command IDs for the Groups Cluster.....	3-36
1132	Table 3-38. Generated Command IDs for the Groups Cluster .....	3-39
1133	Table 3-39. Scenes Attribute Sets .....	3-43
1134	Table 3-40. Scene Management Information Attribute Set.....	3-43
1135	Table 3-41. Fields of a Scene Table Entry.....	3-44
1136	Table 3-42. Received Command IDs for the Scenes Cluster .....	3-45

1137	Table 3-43. Generated Command IDs for the Scenes Cluster .....	3-51
1138	Table 3-44. Values of the Status Field of the Copy Scene Response Command.....	3-55
1139	Table 3-45. Attributes of the On/Off Server Cluster .....	3-57
1140	Table 3-46. Values of the StartUpOnOff Attribute .....	3-59
1141	Table 3-47. Command IDs for the On/Off Cluster .....	3-59
1142	Table 3-48. Values of the Effect Identifier Field of the Off With Effect Command .....	3-60
1143	Table 3-49. Values of the Effect Variant Field of the Off With Effect Command .....	3-60
1144	Table 3-50. On/Off Switch Configuration Attribute Sets .....	3-64
1145	Table 3-51. Attributes of the Switch Information Attribute Set .....	3-65
1146	Table 3-52. Values of the SwitchType Attribute.....	3-65
1147	Table 3-53. Attributes of the Switch Settings Attribute Set.....	3-65
1148	Table 3-54. Values of the SwitchActions Attribute.....	3-66
1149	Table 3-55. Actions on Receipt for On/Off Commands, when Associated with Level Control .....	3-67
1150	Table 3-56. Attributes of the Level Control Server Cluster .....	3-68
1151	Table 3-57. Options Attribute .....	3-70
1152	Table 3-58. Values of the StartUpCurrentLevel attribute .....	3-71
1153	Table 3-59. Command IDs for the Level Control Cluster.....	3-71
1154	Table 3-60. Values of the Move Mode Field .....	3-73
1155	Table 3-61. Actions on Receipt for Move Command.....	3-73
1156	Table 3-62. Values of the Step Mode Field .....	3-74
1157	Table 3-63. Actions on Receipt for Step Command .....	3-74
1158	Table 3-64. Alarms Cluster Attribute Sets .....	3-77
1159	Table 3-65. Attributes of the Alarm Information Attribute Set .....	3-77
1160	Table 3-66. Format of the Alarm Table .....	3-78
1161	Table 3-67. Received Command IDs for the Alarms Cluster.....	3-78
1162	Table 3-68. Generated Command IDs for the Alarms Cluster .....	3-79
1163	Table 3-69. Attributes of the Time Server Cluster .....	3-81
1164	Table 3-70. Bit Values of the TimeStatus Attribute .....	3-82
1165	Table 3-71. Location Attribute Sets.....	3-86
1166	Table 3-72. Attributes of the Location Information Attribute Set .....	3-86
1167	Table 3-73. Bit Values of the LocationType Attribute .....	3-86
1168	Table 3-74. Values of the LocationMethod Attribute.....	3-87
1169	Table 3-75. Attributes of the Location Settings Attribute Set .....	3-88
1170	Table 3-76. Received Command IDs for the Location Cluster .....	3-89
1171	Table 3-77. Generated Command IDs for the RSSI Location Cluster.....	3-94
1172	Table 3-78. Attributes of the Analog Input (Basic) Server Cluster .....	3-100
1173	Table 3-79. Attributes of the Analog Output (Basic) Server Cluster.....	3-101
1174	Table 3-80. Attributes of the Analog Value (Basic) Server Cluster .....	3-103
1175	Table 3-81. Attributes of the Binary Input (Basic) Server Cluster .....	3-105
1176	Table 3-82. Attributes of the Binary Output (Basic) Server Cluster.....	3-106
1177	Table 3-83. Attributes of the Binary Value (Basic) Server Cluster .....	3-107
1178	Table 3-84. Attributes of the Multistate Input (Basic) Server Cluster .....	3-109
1179	Table 3-85. Attributes of the Multistate Output (Basic) Server Cluster.....	3-110
1180	Table 3-86. Attributes of the Multistate Value (Basic) Server Cluster .....	3-112
1181	Table 3-87. AI Types, Type = 0x00: Temperature in Degrees C .....	3-117
1182	Table 3-88. AI Types, Type = 0x01: Relative Humidity in % .....	3-119
1183	Table 3-89. AI Types, Type = 0x02: Pressure in Pascal .....	3-120
1184	Table 3-90. AI Types, Type = 0x03: Flow in Liters/Second.....	3-121
1185	Table 3-91. AI Types, Type = 0x04: Percentage % .....	3-122
1186	Table 3-92. AI types, Type = 0x05: Parts per Million PPM .....	3-122
1187	Table 3-93. AI Types, Type = 0x06: Rotational Speed in RPM.....	3-123
1188	Table 3-94. AI Types, Type = 0x07: Current in Amps.....	3-123
1189	Table 3-95. AI Types, Type = 0x08: Frequency in Hz .....	3-123
1190	Table 3-96. AI Types, Type = 0x09: Power in Watts .....	3-124
1191	Table 3-97. AI Types, Type = 0x0A: Power in kW.....	3-124
1192	Table 3-98. AI Types, Type = 0x0B: Energy in kWh.....	3-124

1193	Table 3-99. AI Types, Type = 0x0C: Count - Unitless.....	3-124
1194	Table 3-100. AI Types, Type = 0x0D: Enthalpy in KJoules/Kg.....	3-125
1195	Table 3-101. AI types, Type = 0x0E: Time in Seconds.....	3-125
1196	Table 3-102. AO Types, Type = 0x00: Temperature in Degrees C.....	3-125
1197	Table 3-103. AO Types, Type = 0x01: Relative Humidity in %.....	3-126
1198	Table 3-104. AO Types, Type = 0x02: Pressure Pascal.....	3-126
1199	Table 3-105. AO Types, Type = 0x03: Flow in Liters/Second.....	3-126
1200	Table 3-106. AO Types, Type = 0x04: Percentage %.....	3-127
1201	Table 3-107. AO Types, Type = 0x05: Parts per Million PPM.....	3-129
1202	Table 3-108. AO Types, Type = 0x06: Rotational Speed RPM.....	3-129
1203	Table 3-109. AO Types, Type = 0x07: Current in Amps.....	3-129
1204	Table 3-110. AO Types, Type = 0x08: Frequency in Hz.....	3-129
1205	Table 3-111. AO Types, Type = 0x09: Power in Watts.....	3-130
1206	Table 3-112. AO Types, Type = 0x0A: Power in kW.....	3-130
1207	Table 3-113. AO Types, Type = 0x0B: Energy in kWh.....	3-130
1208	Table 3-114. AO Types, Type = 0x0C: Count - Unitless.....	3-130
1209	Table 3-115. AO Types, Type = 0x0D: Enthalpy in KJoules/Kg.....	3-131
1210	Table 3-116. AO Types, Type = 0x0E: Time in Seconds.....	3-131
1211	Table 3-117. AV Types, Type = 0x00: Temperature in Degrees C.....	3-131
1212	Table 3-118. AV Types, Type = 0x01: Area in Square Metres.....	3-132
1213	Table 3-119. AV Types, Type = 0x02: Multiplier - Number.....	3-132
1214	Table 3-120. AV Types, Type = 0x03: Flow in Litres/Second.....	3-133
1215	Table 3-121. BI Types, Type = 0x00: Application Domain HVAC.....	3-133
1216	Table 3-122. BI Types, Type = 0x01: Application Domain Security.....	3-139
1217	Table 3-123. BO Types, Type = 0x00: Application Domain HVAC.....	3-140
1218	Table 3-124. BO Types, Type = 0x02: Application Domain Security.....	3-145
1219	Table 3-125. BV Types, Type = 0x00.....	3-146
1220	Table 3-126. MI Types, Type = 0x00: Application Domain HVAC.....	3-146
1221	Table 3-127. MO Types, Type = 0x00: Application Domain HVAC.....	3-147
1222	Table 3-128. MV Types, Type = 0x00: Application Domain HVAC.....	3-148
1223	Table 3-129. Server Attribute Sets of the Diagnostics Cluster.....	3-150
1224	Table 3-130. Hardware Information Attribute Set.....	3-150
1225	Table 3-131. Stack / Network Information Attribute Set.....	3-150
1226	Table 3-132. Server Attributes.....	3-155
1227	Table 3-133. Commands Generated by the Poll Control Server.....	3-158
1228	Table 3-134. Commands Generated by the Poll Control Client.....	3-158
1229	Table 3-135. Attributes of the Power Profile Cluster.....	3-165
1230	Table 3-136. EnergyRemote Attribute.....	3-166
1231	Table 3-137. ScheduleMode Attribute.....	3-166
1232	Table 3-138. Cluster-Specific Commands Received by the Server.....	3-166
1233	Table 3-139. Cluster-Specific Commands Sent by the Server.....	3-174
1234	Table 3-140. PowerProfileState Enumeration Field.....	3-178
1235	Table 3-141. Options Field.....	3-184
1236	Table 3-142. Attributes of the Meter Identification Server Cluster.....	3-190
1237	Table 3-143. Meter Type IDs.....	3-190
1238	Table 3-144. Data Quality IDs.....	3-191
1239	Table 3-145. Attributes of the Level Control for Lighting server cluster.....	3-193
1240	Table 3-146. Options Attribute.....	3-194
1241	Table 3-147. Commands for the Pulse Width Modulation cluster.....	3-194
1242	Table 3-148. Lighting Device State Change.....	3-195
1243	Table 3-149. Attributes of the Pulse Width Modulation server cluster.....	3-197
1244	Table 3-150. Commands for the Pulse Width Modulation cluster.....	3-198
1245	Table 4-1. Illuminance Measurement and Level Sensing Clusters.....	4-1
1246	Table 4-2. Pressure and Flow Measurement Clusters.....	4-2
1247	Table 4-3. Occupancy Sensing Clusters.....	4-3
1248	Table 4-4. Electrical Measurement Clusters.....	4-4

1249	Table 4-5. Illuminance Measurement Attributes .....	4-5
1250	Table 4-6. Values of the LightSensorType Attribute.....	4-6
1251	Table 4-7. Illuminance Level Sensing Attribute Sets .....	4-8
1252	Table 4-8. Illuminance Level Sensing Information Attribute Set.....	4-8
1253	Table 4-9. Values of the LevelStatus Attribute .....	4-8
1254	Table 4-10. Values of the LightSensorType Attribute.....	4-8
1255	Table 4-11. Illuminance Level Sensing Settings Attribute Set.....	4-9
1256	Table 4-12. Temperature Measurement Attribute Sets .....	4-10
1257	Table 4-13. Temperature Measurement Information Attribute Set .....	4-11
1258	Table 4-14. Pressure Measurement Attribute Sets.....	4-12
1259	Table 4-15. Pressure Measurement Information Attribute Set .....	4-13
1260	Table 4-16. Extended Pressure Measurement Information Attribute Set.....	4-14
1261	Table 4-17. Flow Measurement Attribute Sets.....	4-16
1262	Table 4-18. Flow Measurement Information Attribute Set .....	4-16
1263	Table 4-19. Attributes of the Water Content cluster.....	4-18
1264	Table 4-20. Occupancy Sensor Attribute Sets.....	4-20
1265	Table 4-21. Occupancy Sensor Information Attribute Set .....	4-20
1266	Table 4-22. Values of the OccupancySensorType Attribute .....	4-20
1267	Table 4-23. The OccupancySensorTypeBitmap Attribute .....	4-21
1268	Table 4-24. Mapping between OccupancySensorType and OccupancySensorTypeBitmap Attributes .....	4-21
1269	Table 4-25. Attributes of the PIR Configuration Attribute Set.....	4-21
1270	Table 4-26. Attributes of the Ultrasonic Configuration Attribute Set.....	4-22
1271	Table 4-27. Attributes of the Physical Contact Configuration Attribute Set.....	4-22
1272	Table 4-28. Attributes of the Electrical Measurement Cluster .....	4-24
1273	Table 4-29. Electrical Measurement Cluster Basic Information.....	4-25
1274	Table 4-30. MeasurementType Attribute.....	4-25
1275	Table 4-31. DC Measurement Attributes.....	4-26
1276	Table 4-32. DC Formatting Attributes.....	4-27
1277	Table 4-33. AC (Non-phase Specific) Measurement Attributes.....	4-28
1278	Table 4-34. AC (Non-phase Specific) Formatting Attributes.....	4-30
1279	Table 4-35. AC (Single Phase or Phase A) Measurement Attributes .....	4-31
1280	Table 4-36. AC Formatting Attributes.....	4-33
1281	Table 4-37. DC Manufacturer Threshold Alarms Attributes.....	4-34
1282	Table 4-38. AC Manufacturer Threshold Alarms Attributes.....	4-35
1283	Table 4-39. AC Phase B Measurements Attributes .....	4-37
1284	Table 4-40. AC Phase C Measurements Attributes .....	4-40
1285	Table 4-41. Generated Command ID's for the Electrical Measurement Server .....	4-43
1286	Table 4-42. List of Status Valid Values .....	4-44
1287	Table 4-43. Generated Command IDs for the Electrical Measurement Client .....	4-45
1288	Table 4-44. Attributes of the Electrical Conductivity Measurement server cluster .....	4-46
1289	Table 4-45. Attributes of the pH Measurement server cluster.....	4-48
1290	Table 4-46. Attributes of the Wind Speed Measurement server cluster.....	4-49
1291	Table 4-47. Attributes of the Concentration Measurement server cluster .....	4-53
1292	Table 5-1. Clusters Specified for the Lighting Functional Domain .....	5-1
1293	Table 5-2. Hue Control Attribute Sets .....	5-3
1294	Table 5.3. Attributes of the Color Information Attribute Set .....	5-4
1295	Table 5-4. Values of the DriftCompensation Attribute.....	5-6
1296	Table 5-5. Values of the ColorMode Attribute.....	5-7
1297	Table 5-6. Options Attribute .....	5-8
1298	Table 5-7. Values of the EnhancedColorMode Attribute.....	5-8
1299	Table 5-8. Bit Values of the ColorCapabilities Attribute.....	5-9
1300	Table 5-9. Values of the StartUpColorTemperatureMireds attribute.....	5-11
1301	Table 5-10. Defined Primaries Information Attribute Set.....	5-11
1302	Table 5-11. Additional Defined Primaries Information Attribute Set.....	5-12
1303	Table 5-12. Defined Color Points Settings Attribute Set .....	5-13
1304	Table 5-13. Command IDs for the Color Control Cluster.....	5-14

1305	Table 5-14. Values of the Direction Field.....	5-16
1306	Table 5-15. Values of the Move Mode Field .....	5-17
1307	Table 5-16. Actions on Receipt for Move Hue Command.....	5-18
1308	Table 5-17. Values of the Step Mode Field .....	5-18
1309	Table 5-18. Actions on Receipt for Step Hue Command.....	5-19
1310	Table 5-19. Values of the Move Mode Field .....	5-20
1311	Table 5-20. Actions on Receipt for Move Saturation Command .....	5-21
1312	Table 5-21. Values of the Step Mode Field .....	5-21
1313	Table 5-22. Actions on Receipt for Step Saturation Command.....	5-22
1314	Table 5-23. Actions on Receipt of the Enhanced Move Hue Command .....	5-26
1315	Table 5-24. Actions on Receipt for the Enhanced Step Hue Command .....	5-28
1316	Table 5-25. Values of the Action Field of the Color Loop Set Command .....	5-29
1317	Table 5-26. Values of the Direction Field of the Color Loop Set Command .....	5-30
1318	Table 5-27. Actions on Receipt of the Move Color Temperature Command.....	5-32
1319	Table 5-28. Actions on Receipt of the Step Color Temperature Command.....	5-34
1320	Table 5-29. Ballast Configuration Attribute Sets .....	5-36
1321	Table 5-30. Attributes of the Ballast Information Attribute Set .....	5-36
1322	Table 5-31. Bit Usage of the BallastStatus Attribute.....	5-37
1323	Table 5-32. Attributes of the Ballast Settings Attribute Set .....	5-37
1324	Table 5-33. Attributes of the Lamp Information Attribute Set.....	5-38
1325	Table 5-34. Attributes of the Lamp Settings Attribute Set.....	5-38
1326	Table 5-35. Values of the MainsAlarmMode Attribute .....	5-39
1327	Table 5-36. Examples of The Dimming Light Curve .....	5-40
1328	Table 6-1. Clusters Specified in the HVAC Functional Domain.....	6-1
1329	Table 6-2. Pump Configuration Attribute Sets.....	6-4
1330	Table 6-3. Attributes of the Pump Information Attribute Set.....	6-4
1331	Table 6-4. Attributes of the Pump Dynamic Information Attribute Set .....	6-7
1332	Table 6-5. Values of the PumpStatus Attribute.....	6-7
1333	Table 6-6. Attributes of the Pump Settings Attribute Set.....	6-9
1334	Table 6-7. Values of the OperationMode Attribute .....	6-11
1335	Table 6-8. Values of the ControlMode Attribute.....	6-11
1336	Table 6-9. Alarm Codes.....	6-12
1337	Table 6-10. Currently Defined Thermostat Attribute Sets .....	6-14
1338	Table 6-11. Attributes of the Thermostat Information Attribute Set .....	6-14
1339	Table 6-12. HVAC System Type Configuration Values.....	6-16
1340	Table 6-13. Attributes of the Thermostat Settings Attribute Set .....	6-17
1341	Table 6-14. RemoteSensing Attribute Bit Values .....	6-19
1342	Table 6-15. ControlSequenceOfOperation Attribute Values.....	6-19
1343	Table 6-16. SystemMode Attribute Values.....	6-20
1344	Table 6-17. Interpretation of SystemMode Values.....	6-20
1345	Table 6-18. Alarm Codes.....	6-20
1346	Table 6.19 Thermostat Running Mode Attribute Values .....	6-21
1347	Table 6-20. Thermostat Schedule & HVAC Relay Attribute Set .....	6-21
1348	Table 6-21. StartofWeek Enumeration Values.....	6-21
1349	Table 6-22. TemperatureSetpointHold Attribute Values .....	6-22
1350	Table 6-23. ThermostatProgrammingOperationMode Attribute Values .....	6-23
1351	Table 6-24. HVAC Relay State Values .....	6-23
1352	Table 6-25. Thermostat Setpoint Change Tracking Attribute Set.....	6-23
1353	Table 6-26. SetpointChangeSource Values.....	6-24
1354	Table 6-27. SetpointChangeAmount Values.....	6-24
1355	Table 6-28. Attributes of the AC Information Attribute Set .....	6-27
1356	Table 6-29. ACType Enumeration .....	6-27
1357	Table 6-30. ACRefrigerantType Enumeration .....	6-27
1358	Table 6-31. ACCompressorType Enumeration .....	6-28
1359	Table 6-32. ACErrorCode Values .....	6-28
1360	Table 6-33. ACLouverPosition Values.....	6-28



1361	Table 6-34. ACCapacity Enumeration.....	6-29
1362	Table 6-35. Command IDs for the Thermostat Cluster.....	6-29
1363	Table 6-36. Mode Field Values for Setpoint Raise/Lower Command.....	6-30
1364	Table 6-37. Day Of Week for Sequence Values.....	6-31
1365	Table 6-38. Mode for Sequence Values.....	6-31
1366	Table 6-39. Server Commands Send Command ID.....	6-35
1367	Table 6-40. Attributes of the Fan Control Cluster.....	6-37
1368	Table 6-41. FanMode Attribute Values.....	6-38
1369	Table 6-42. FanSequenceOperation Attribute Values.....	6-38
1370	Table 6-43. Dehumidification Control Attribute Sets.....	6-39
1371	Table 6-44. Dehumidification Information Attribute Set.....	6-40
1372	Table 6-45. Dehumidification Settings Attribute Set.....	6-40
1373	Table 6-46. RelativeHumidityMode Attribute Values.....	6-40
1374	Table 6-47. DehumidificationLockout Attribute Values.....	6-41
1375	Table 6-48. RelativeHumidityMode Attribute Values.....	6-41
1376	Table 6-49. Thermostat User Interface Configuration Cluster.....	6-42
1377	Table 6-50. DisplayMode Attribute Values.....	6-43
1378	Table 6-51. KeypadLockout Attribute Values.....	6-43
1379	Table 6-52. ScheduleProgrammingVisibility Attribute Values.....	6-43
1380	Table 7-1. Clusters Specified in the Closures Functional Domain.....	7-1
1381	Table 7-2. Shade Configuration Attribute Sets.....	7-3
1382	Table 7-3. Attributes of the Shade Information Attribute Set.....	7-3
1383	Table 7-4. Bit Values for the Status Attribute.....	7-4
1384	Table 7-5. Attributes of the Shade Settings Attribute Set.....	7-4
1385	Table 7-6. Values of the Mode Attribute.....	7-4
1386	Table 7-7. Attribute Sets Description.....	7-8
1387	Table 7-8. Current Information Attribute Set.....	7-8
1388	Table 7-9. LockState Attribute Values.....	7-8
1389	Table 7-10. LockType Attribute Values.....	7-9
1390	Table 7-11. ActuatorEnabled Attribute Values.....	7-9
1391	Table 7-12. DoorState Attribute Values.....	7-10
1392	Table 7-13. User, PIN, Schedule, Log Information Attribute Set.....	7-10
1393	Table 7-14. Operational Settings Attribute Set.....	7-12
1394	Table 7-15. Operating Modes.....	7-13
1395	Table 7-16. Bit Values for the SupportedOperatingModes Attribute.....	7-14
1396	Table 7-17. Modes for the LEDSettings Attribute.....	7-14
1397	Table 7-18. Settings for the SoundVolume Attribute.....	7-14
1398	Table 7-19. DefaultConfigurationRegister Attribute.....	7-15
1399	Table 7-20. Security Settings Attribute Set.....	7-16
1400	Table 7-21. Alarm and Event Masks Attribute Set.....	7-17
1401	Table 7-22. Alarm Code Table.....	7-17
1402	Table 7-23. Commands Received by the Server Cluster.....	7-19
1403	Table 7-24. User Status Value.....	7-21
1404	Table 7-25. User Type Value.....	7-22
1405	Table 7-26. User Status Byte Values for Set RFID Code Command.....	7-28
1406	Table 7-27. Commands Generated by the Server Cluster.....	7-29
1407	Table 7-28. Operation Event Source Value.....	7-41
1408	Table 7-29. Operation Event Code Value.....	7-42
1409	Table 7-30. Keypad Operation Event Value.....	7-43
1410	Table 7-31. RF Operation Event Value.....	7-43
1411	Table 7-32. Manual Operation Event Value.....	7-44
1412	Table 7-33. RFID Operation Event Value.....	7-44
1413	Table 7-34. Operation Event Source Value.....	7-45
1414	Table 7-35. Programming Event Codes.....	7-46
1415	Table 7-36. Keypad Programming Event Value.....	7-47
1416	Table 7-37. RF Programming Event Value.....	7-47

1417	Table 7-38. RFID Programming Event Value.....	7-48
1418	Table 7-39. Window Covering Attribute Set .....	7-50
1419	Table 7-40. Window Covering Information Attribute Set .....	7-50
1420	Table 7-41. Window Covering Type .....	7-51
1421	Table 7-42. Bit Meanings for the Config/Status Attribute .....	7-52
1422	Table 7-43. Window Covering Settings Attribute Set .....	7-53
1423	Table 7-44. Bit Meanings for the Mode Attribute .....	7-54
1424	Table 7-45. Commands Received by the Window Covering Server Cluster.....	7-55
1425	Table 8-1. Clusters of the Security and Safety Functional Domain .....	8-1
1426	Table 8-2. Attribute Sets for the IAS Zone Cluster .....	8-3
1427	Table 8-3. Attributes of the Zone Information Attribute Set.....	8-3
1428	Table 8-4. Values of the ZoneState Attribute.....	8-3
1429	Table 8-5. Values of the ZoneType Attribute .....	8-4
1430	Table 8-6. Values of the ZoneStatus Attribute.....	8-4
1431	Table 8-7. Usage of alarm bits of ZoneStatus Attribute for door/window handle zone type (0x0016).....	8-5
1432	Table 8-8. Attributes of the Zone Settings Attribute Set.....	8-6
1433	Table 8-9. Received Command IDs for the IAS Zone Cluster .....	8-8
1434	Table 8-10. Values of the Enroll Response Code.....	8-9
1435	Table 8-11. Generated Command IDs for the IAS Zone Cluster.....	8-11
1436	Table 8-12. Format of the Zone Table.....	8-14
1437	Table 8-13. Received Command IDs for the IAS ACE Cluster .....	8-14
1438	Table 8-14. Arm Mode Field Values.....	8-15
1439	Table 8-15. Generated Command IDs for the IAS ACE Cluster.....	8-19
1440	Table 8-16. Arm Notification Values .....	8-19
1441	Table 8-17. PanelStatus Field Values .....	8-22
1442	Table 8-18. Values of Bypass Result Field.....	8-26
1443	Table 8-19. Attributes of the IAS WD (Server) Cluster .....	8-28
1444	Table 8-20. Received Command IDs for the IAS WD Server Cluster.....	8-28
1445	Table 8-21. Warning Modes .....	8-29
1446	Table 8-22. Values of the Strobe Field .....	8-29
1447	Table 8-23. Siren Level Field Values .....	8-30
1448	Table 8-24. Strobe Level Field Values .....	8-30
1449	Table 8-25. Squawk Mode Field .....	8-31
1450	Table 8-26. Strobe Bit .....	8-31
1451	Table 8-27. Squawk Level Field Values.....	8-31
1452	Table 9-1. Clusters of the Protocol Interfaces Functional.....	9-1
1453	Table 9-2. Attributes of the Generic Tunnel Cluster .....	9-3
1454	Table 9-3. Command IDs Received by the Generic Tunnel Cluster.....	9-4
1455	Table 9-4. Command IDs Generated by the Generic Tunnel Cluster .....	9-5
1456	Table 9-5. Command IDs for the BACnet Protocol Tunnel Cluster.....	9-7
1457	Table 9-6. Attributes of the Analog Input (BACnet Regular) Server .....	9-9
1458	Table 9-7. Attributes of the Analog Input (BACnet Extended) Server.....	9-10
1459	Table 9-8. Attributes of the Analog Output (BACnet Regular) Server.....	9-11
1460	Table 9-9. Attributes of the Analog Output (BACnet Extended) Server .....	9-13
1461	Table 9-10. Attributes of the Analog Value (BACnet Regular) Server .....	9-14
1462	Table 9-11. Attributes of the Analog Value (BACnet Extended) Server .....	9-15
1463	Table 9-12. Attributes of the Binary Input (BACnet Regular) Server .....	9-16
1464	Table 9-13. Attributes of the Binary Input (BACnet Extended) Server .....	9-18
1465	Table 9-14. Attributes of the Binary Output (BACnet Regular) Server.....	9-19
1466	Table 9-15. Attributes of the Binary Output (BACnet Extended) Server .....	9-21
1467	Table 9-16. Attributes of the Binary Value (BACnet Regular) Server .....	9-22
1468	Table 9-17. Attributes of the Binary Value (BACnet Extended) Server.....	9-23
1469	Table 9-18. Attributes of the Multistate Input (BACnet Regular) Server .....	9-25
1470	Table 9-19. Attributes of Multistate Input (BACnet Extended) Server .....	9-26
1471	Table 9-20. Attributes of Multistate Output (BACnet Regular) Server .....	9-27
1472	Table 9-21. Attributes of Multistate Output (BACnet Extended) Server.....	9-28

1473	Table 9-22. Attributes of Multistate Value (BACnet Regular) Server.....	9-30
1474	Table 9-23. Attributes of Multistate Value (BACnet Extended) Server .....	9-31
1475	Table 9-24. Definitions Used in ISO 7816 Protocol Tunnel Description.....	9-35
1476	Table 9-25. Attributes for the ISO7816 Tunnel Cluster.....	9-37
1477	Table 9-26. Status Values .....	9-37
1478	Table 9-27. Received Command IDs for the ISO7816 Tunnel Cluster .....	9-37
1479	Table 9-28. Generated Command IDs for the ISO7816 Tunnel Cluster .....	9-38
1480	Table 9-29. Attributes of the Partition Cluster .....	9-42
1481	Table 9-30. Server Received Command IDs for the Partition Cluster.....	9-44
1482	Table 9-31. Generated Command IDs for the Partition Cluster .....	9-46
1483	Table 9-32. Registration Table of Clusters Using the Partition Cluster.....	9-48
1484	Table 9-33 – Attributes of the 11073 Protocol Tunnel server cluster .....	9-51
1485	Table 9-34 – Command IDs for the 11073 protocol tunnel cluster .....	9-52
1486	Table 9-35 – Connect status values .....	9-56
1487	Table 10-1. Smart Energy Clusters.....	10-1
1488	Table 10-2. Price Cluster Attribute Sets.....	10-3
1489	Table 10-3. Tier Label Attribute Set.....	10-3
1490	Table 10-4. Block Threshold Attribute Set.....	10-4
1491	Table 10-5. Block Period Attribute Set.....	10-5
1492	Table 10-6. Commodity Attribute Set .....	10-6
1493	Table 10-7. Values and Descriptions for the CalorificValueUnit Attribute.....	10-7
1494	Table 10-8. Block Price Information Attribute Set.....	10-7
1495	Table 10-9. Billing Information Attribute Set .....	10-10
1496	Table 10-10. Received Command IDs for the Price Cluster.....	10-11
1497	Table 10-11. Generated Command IDs for the Price Cluster .....	10-15
1498	Table 10-12. Price Tier Sub-field Enumerations .....	10-17
1499	Table 10-13. Register Tier Sub-field Enumerations .....	10-18
1500	Table 10-14. Alternate Cost Unit Enumerations .....	10-19
1501	Table 10-15. Price Control Field BitMap.....	10-20
1502	Table 10-16. Block Period Control Field BitMap.....	10-21
1503	Table 10-17. Price Client Cluster Attributes .....	10-23
1504	Table 10-18. Command IDs for the Demand Response and Load Control Server.....	10-27
1505	Table 10-19. Device Class Field BitMap/Encoding .....	10-28
1506	Table 10-20. Criticality Levels.....	10-29
1507	Table 10-21. Event Control Field BitMap.....	10-31
1508	Table 10-22. Cancel Control .....	10-32
1509	Table 10-23. Format of the Cancel All Load Control Events Command Payload.....	10-33
1510	Table 10-24. Cancel All Command Cancel Control Field .....	10-33
1511	Table 10-25. Demand Response Client Cluster Attributes.....	10-34
1512	Table 10-26. Generated Command IDs for the Demand Response and Load Control Client.....	10-35
1513	Table 10-27. Event Status Field Values .....	10-36
1514	Table 10-28. Enumerated Values of Signature Types .....	10-37
1515	Table 10-29. Metering Cluster Attribute Sets .....	10-53
1516	Table 10-30. Reading Information Attribute Set .....	10-54
1517	Table 10-31. Block Enumerations.....	10-57
1518	Table 10-32. Supply Status Attribute Enumerations .....	10-58
1519	Table 10-33. TOU Information Attribute Set.....	10-59
1520	Table 10-34. Meter Status Attribute Set .....	10-62
1521	Table 10-35. Mapping of the Status Attribute (Electricity).....	10-62
1522	Table 10-36. Meter Status Attribute (Gas).....	10-63
1523	Table 10-37. Meter Status Attribute (Water).....	10-63
1524	Table 10-38. Meter Status Attribute (Heat and Cooling).....	10-63
1525	Table 10-39. Formatting Examples .....	10-65
1526	Table 10-40. Formatting Attribute Set.....	10-65
1527	Table 10-41. UnitofMeasure Attribute Enumerations.....	10-66

1528	Table 10-42. MeteringDeviceType Attribute .....	10-68
1529	Table 10-43. TemperatureUnitOfMeasure Enumeration .....	10-70
1530	Table 10-44. Historical Consumption Attribute Set.....	10-71
1531	Table 10-45. Load Profile Configuration Attribute Set .....	10-74
1532	Table 10-46. Supply Limit Attribute Set .....	10-74
1533	Table 10-47. Block Information Attribute Set .....	10-75
1534	Table 10-48. Alarm Attribute Set .....	10-79
1535	Table 10-49. Alarm Code Groups .....	10-79
1536	Table 10-50. Generic Alarm Group.....	10-80
1537	Table 10-51. Electricity Alarm Group .....	10-80
1538	Table 10-52. Generic Flow/Pressure Alarm Group .....	10-81
1539	Table 10-53. Water Specific Alarm Group .....	10-81
1540	Table 10-54. Heat and Cooling Specific Alarm Group.....	10-82
1541	Table 10-55. Gas Specific Alarm Group.....	10-82
1542	Table 10-56. Generated Command IDs for the Metering Server.....	10-82
1543	Table 10-57. Status Field Values.....	10-83
1544	Table 10-58. ProfileIntervalPeriod Timeframes .....	10-83
1545	Table 10-59. Generated Command IDs for the Metering Client .....	10-85
1546	Table 10-60. Interval Channel Values .....	10-85
1547	Table 10-61. Generated Command IDs for the Messaging Server.....	10-90
1548	Table 10-62. Message Control Field Bit Map .....	10-90
1549	Table 10-63. Messaging Client Commands .....	10-92
1550	Table 10-64. Tunneling Cluster Attributes .....	10-99
1551	Table 10-65. Cluster Parameters Passed Through Commands.....	10-99
1552	Table 10-66. Cluster-specific Commands Received by the Server.....	10-100
1553	Table 10-67. ProtocolID Enumerations .....	10-100
1554	Table 10-68. TransferDataStatus Values .....	10-103
1555	Table 10-69. Cluster-Specific Commands Sent by the Server .....	10-105
1556	Table 10-70. TunnelStatus Values .....	10-106
1557	Table 10-71. Clusters Specified for the Secure Communication Functional Domain .....	10-112
1558	Table 10-72. Key Establishment Attribute Sets .....	10-115
1559	Table 10-73. Information Attribute Sets .....	10-115
1560	Table 10-74. Values of the KeyEstablishmentSuite Attribute.....	10-116
1561	Table 10-75. Received Command IDs for the Key Establishment Cluster Server .....	10-116
1562	Table 10-76. Terminate Key Establishment Command Status Field .....	10-118
1563	Table 10-77. Key Establishment Attribute Sets .....	10-119
1564	Table 10-78. Attributes of the Information Attribute Set.....	10-119
1565	Table 10-79. Values of the KeyEstablishmentSuite Attribute.....	10-119
1566	Table 10-80. Received Command IDs for the Key Establishment Cluster Client.....	10-120
1567	Table 10-81. Terminate Key Establishment Command Status Field .....	10-122
1568	Table 10-82. Parameters Used by Methods of the CBKE Protocol .....	10-127
1569	Table 11-1. Clusters Specified in This Document .....	11-2
1570	Table 11-2. OTA Header Fields .....	11-6
1571	Table 11-3. OTA Header Field Control Bitmask .....	11-7
1572	Table 11-4. Image Type Values .....	11-8
1573	Table 11-5. Recommended File Version Definition .....	11-8
1574	Table 11-6. ZigBee Stack Version Values .....	11-9
1575	Table 11-7. Security Credential Version.....	11-10
1576	Table 11-8. Hardware Version Format .....	11-10
1577	Table 11-9. Tag Identifiers .....	11-11
1578	Table 11-10. Attributes of OTA Upgrade Cluster .....	11-19
1579	Table 11-11. Image Upgrade Status Attribute Values .....	11-21
1580	Table 11-12. UpgradeActivationPolicy Enumerations .....	11-22
1581	Table 11-13. UpgradeTimeoutPolicy Enumerations .....	11-23
1582	Table 11-14. Parameters of OTA Upgrade Cluster .....	11-23

1583	Table 11-15. Meaning of CurrentTime and UpgradeTime Parameters.....	11-24
1584	Table 11-16. OTA Upgrade Cluster Command Frames .....	11-27
1585	Table 11-17. Status Code Defined and Used by OTA Upgrade Cluster .....	11-28
1586	Table 11-18. Image Notify Command Payload Type .....	11-29
1587	Table 11-19. Query Next Image Request Field Control Bitmask.....	11-31
1588	Table 11-20. Image Block Request Field Control Bitmask.....	11-35
1589	Table 11-21. Image Page Request Field Control Bitmask .....	11-38
1590	Table 12-1. Telecom Cluster List.....	12-1
1591	Table 12-2. Clusters Specified for the Information Delivery .....	12-3
1592	Table 12-3. Information Cluster Attribute Sets .....	12-7
1593	Table 12-4. Node Information Attribute Set .....	12-7
1594	Table 12-5. Contents Information Attribute Set.....	12-8
1595	Table 12-6. Received Command IDs for the Information Cluster.....	12-8
1596	Table 12-7. Inquiry ID.....	12-9
1597	Table 12-8. Data Type IDs.....	12-9
1598	Table 12-9. Preference Type .....	12-12
1599	Table 12-10. Value of the Access Control Field .....	12-14
1600	Table 12-11. Generated Command IDs for the Information Cluster .....	12-18
1601	Table 12-12. Enumerated Status Values Used in the ZCL.....	12-24
1602	Table 12-13. Chatting Attributes Sets.....	12-27
1603	Table 12-14. Attributes of the User Related Attribute Set .....	12-27
1604	Table 12-15. Attributes of Chat Session Related Attribute Set .....	12-28
1605	Table 12-16. Command IDs for the Chatting Cluster.....	12-28
1606	Table 12-17. Generated Command IDs for the Chatting Cluster .....	12-32
1607	Table 12-18. VoZ Attribute Sets .....	12-38
1608	Table 12-19. Attributes of the Voice Information Attribute Set.....	12-38
1609	Table 12-20. Command IDs for the VoZ Cluster .....	12-40
1610	Table 12-21. Generated Command IDs for the VoZ Cluster.....	12-43
1611	Table 12-22. The Error Flag of Voice Transmission Response.....	12-44
1612	Table 13-1. Commissioning Attribute Sets .....	13-3
1613	Table 13-2. Attributes of the Startup Parameters Attribute Set.....	13-3
1614	Table 13-3. Stack Profile Compatibility for the ShortAddress Attribute .....	13-5
1615	Table 13-4. Stack Profile Compatibility for the PANId Attribute .....	13-5
1616	Table 13-5. StartupControl Attribute Usage .....	13-7
1617	Table 13-6. Stack Profile Compatibility for the StartupControl Attribute .....	13-8
1618	Table 13-7. Attributes of the Join Parameters Attribute Set.....	13-9
1619	Table 13-8. Attributes of the End Device Parameters Attribute Set .....	13-10
1620	Table 13-9. Attributes of the Concentrator Parameters Attribute Set .....	13-11
1621	Table 13-10. Commands Received by the Commissioning Cluster Server.....	13-12
1622	Table 13-11. Startup Mode Sub-field Values.....	13-13
1623	Table 13-12. Commands Generated by the Commissioning Cluster Server .....	13-16
1624	Table 13-13. Commands Received by the Server Side of the Touchlink Commissioning Cluster .....	13-20
1625	Table 13-14. Values of the Identify Duration Field.....	13-23
1626	Table 13-15. Commands Generated by the Server Side of the Touchlink Commissioning Cluster .....	13-32
1627	Table 13-16. Values of the Status Field of the Network Start Response Command Frame .....	13-38
1628	Table 13-17. Values of the Status Field of the Network Join Router Response Command Frame.....	13-39
1629	Table 13-18. Values of the Status Field of the Network Join End Device Response Command Frame ..	13-41
1630	Table 13-19. Commands Received by the Client Side of the ZLL Commissioning Cluster .....	13-44
1631	Table 13-20. Commands Generated by the Client Side of the ZLL Commissioning Cluster.....	13-44
1632	Table 13-21. Touchlink Commissioning Constants.....	13-45
1633	Table 13-22. Touchlink Commissioning Attributes .....	13-45
1634	Table 13-23. Key Encryption Algorithms.....	13-53
1635	Table 14-1. Clusters Specified in this Chapter .....	14-1
1636	Table 14-2. Attributes of the Retail Tunnel cluster .....	14-3
1637	Table 14-3. Cluster-specific Commands Received by the Server.....	14-3
1638	Table 14-4. Attributes of the Mobile Device Cleaning Cluster .....	14-6

1639	Table 14-5. Cluster-specific Commands Generated by the Server .....	14-6
1640	Table 14-6. Attributes of the Neighbor Cleaning Cluster .....	14-9
1641	Table 14-7. Cluster-specific Commands Generated by the Server .....	14-9
1642	Table 14-8. Attributes of the Nearest Gateway Cluster .....	14-11
1643	Table 15-1. Appliance Management Clusters .....	15-1
1644	Table 15-2. Appliance Control Attribute Set .....	15-2
1645	Table 15-3. Attributes of the Appliance Functions Attribute Set .....	15-3
1646	Table 15-4. Time Encoding .....	15-3
1647	Table 15-5. Cluster-specific Commands Received by the Server.....	15-4
1648	Table 15-6. Command Identification Values .....	15-4
1649	Table 15-7. Format of the Event ID Enumerator.....	15-7
1650	Table 15-8. Cluster-specific Commands Sent by the Server.....	15-8
1651	Table 15-9. Appliance Status Values.....	15-9
1652	Table 15-10. Remote Enable Flags Values .....	15-9
1653	Table 15-11. Appliance Identification Attribute Sets .....	15-11
1654	Table 15-12. Attributes of the Appliance Identification Attribute Set.....	15-12
1655	Table 15-13. Basic Appliance Identification Content Specification.....	15-12
1656	Table 15-14. Product Type IDs .....	15-12
1657	Table 15-15. Attributes of the Extended Appliance Identification Attribute Set.....	15-13
1658	Table 15-16. CECED Specification Version.....	15-15
1659	Table 15-17. Received Commands IDs for the Events and Alerts Cluster.....	15-17
1660	Table 15-18. Generated Commands IDs for the Appliance Events and Alerts Cluster.....	15-17
1661	Table 15-19. Alert Count Organization .....	15-18
1662	Table 15-20. Alerts Structure Organization .....	15-19
1663	Table 15-21. Event Identification .....	15-20
1664	Table 15-22. Server Attributes .....	15-23
1665	Table 15-23. Commands Generated by the Appliance Statistics Server .....	15-23
1666	Table 15-24. Commands Generated by the Appliance Statistics Client.....	15-26
1667		

# CHAPTER 1 INTRODUCTION

1668  
 1669 The Zigbee Cluster Library is made of individual chapters such as this one. See Document Control in the Zigbee  
 1670 Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation  
 1671 where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in  
 1672 Chapter 1 and are made using [*Rn*] notation.

## 1.1 Scope and Purpose

1674 This document specifies the Zigbee Cluster Library (ZCL). The ZCL is a repository for cluster functionality that is  
 1675 developed by the Zigbee Alliance, and is a working library with regular updates as new functionality is added.

1676 A developer constructing a new application should use the ZCL to find relevant cluster functionality that can be  
 1677 incorporated into the new application. Correspondingly, new clusters that are defined for applications should be  
 1678 considered for inclusion in the ZCL.

1679 The ZCL consists of the ZCL Foundation, a set of elements that apply across the entire library (such as frame  
 1680 structures, attribute access commands and data types), and a number of sets of clusters. Clusters that are generally  
 1681 useful across many application domains are included in the General set. Clusters that are intended for use mainly in  
 1682 specific application domains are grouped together in domain oriented sets.

## 1.2 Acronyms and Abbreviations

Table 1-1. Acronyms and Abbreviations

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

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



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

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

1685

## 1.3 Definitions

1686 Many of these terms are described in more detail in the core stack specification [Z1], or the Application Architecture  
1687 specification [Z5].

1688 **Application Cluster:** An application cluster generates persistent functional application transactions between client  
1689 and server.

1690 **Attribute:** A data entity which represents a physical quantity or state. This data is communicated to other devices  
1691 using commands.

1692 **Binding:** A persistent mapping of a local cluster instance to one or more corresponding remote cluster instances. A  
1693 binding can be broadcast, groupcast, or unicast. A unicast binding includes an address (IEEE or network) and endpoint.

1694 **Cluster:** A cluster is a specification defining one or more attributes, commands, behaviors and dependencies, that  
1695 supports an independent utility or application function. The term may also be used for an implementation or instance  
1696 of such a specification on an endpoint.

1697 **Cluster identifier:** The cluster identifier is a 16-bit number that maps to (identifies) a single cluster specification.  
1698 More than one cluster identifier may map to a cluster specification, each defining a different scope and purpose.  
1699 Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.

1700 **Client:** A cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint. Typically  
1701 this interface sends commands that manipulate the attributes on the corresponding server cluster. A client cluster  
1702 communicates with a corresponding remote server cluster with the same cluster identifier.

1703 **Corresponding cluster:** The opposite side of a cluster (client to a server, or server to a client).

1704 **Device:** A specification which defines a unique device identifier and a set of mandatory and optional clusters to be  
1705 implemented on a single endpoint. The term may also be used for an implementation or instance of the device  
1706 specification on an endpoint.

1707 **Node:** A Zigbee node (or node) is a single testable implementation of a Zigbee application on a single Zigbee stack,  
1708 with a single network address, on a single network.

1709 **Product:** A product is a node that is intended to be marketed.

1710 **Server:** A cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint. Typically,  
1711 this interface supports all or most of the attributes of the cluster. A server cluster communicates with a corresponding  
1712 remote client cluster with the same cluster identifier.

1713 **Service discovery:** The ability of a device to locate services of interest.

1714 **Sleepy End Device:** A Zigbee End Device with rxOnWhenIdle set to FALSE.

1715 **Utility Cluster:** A utility cluster is not part of the application function of the product. It may be used for  
1716 commissioning, configuration, discovery, addressing, diagnostics, etc.

1717 **Type 1 Cluster:** A type 1 cluster's primary function is to initiate transactions from the client to the server.

1718 **Type 2 Cluster:** A type 2 cluster's primary function is to initiate transactions from the server to the client.

1719 **Zigbee Coordinator:** An IEEE 802.15.4-2003 PAN coordinator.

1720 **Zigbee End Device:** an IEEE 802.15.4-2003 RFD or FFD participating in a Zigbee network, which is neither the  
1721 Zigbee coordinator nor a Zigbee router.

1722 **Zigbee Router:** an IEEE 802.15.4-2003 FFD participating in a Zigbee network, which is not the Zigbee coordinator  
1723 but may act as an IEEE 802.15.4-2003 coordinator within its personal operating space, that is capable of routing  
1724 messages between devices and supporting associations.

## 1725 1.4 Conformance Levels

---

1726 **Expected:** A key word used to describe the behavior of the hardware or software in the design models *assumed* by  
1727 this Draft. Other hardware and software design models may also be implemented.

1728 **May:** A key word that indicates flexibility of choice with *no implied preference*.

1729 **Shall:** A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory  
1730 requirements.

1731 **Should:** A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is*  
1732 *recommended*.

## 1733 1.5 References

---

1734 The following standards and specifications contain provisions, which through reference in this document constitute  
1735 provisions of this specification. All the standards and specifications listed are normative references. At the time of  
1736 publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to  
1737 agreements based on this specification are encouraged to investigate the possibility of applying the most recent  
1738 editions of the standards and specifications indicated below.

### 1739 1.5.1 Zigbee Alliance Documents

- 1740 [Z1] Zigbee 053474, Zigbee Specification
- 1741 [Z2] Zigbee 064321, Zigbee Stack Profile
- 1742 [Z3] Zigbee 074855, Zigbee PRO Stack Profile
- 1743 [Z4] Zigbee 08006, Zigbee-2007 Layer PICs and Stack Profiles
- 1744 [Z5] Zigbee 130589, Application Architecture
- 1745 [Z6] Zigbee 130402, Base Device Behavior Specification
- 1746 [Z7] Zigbee 053298, Profile Identifier Database
- 1747 [Z8] Zigbee 106050, Zigbee Device internetworking, list of Device IDs
- 1748 [Z9] Zigbee 075356, Smart Energy Profile Specification
- 1749 [Z10] Zigbee 03084, Zigbee Key Establishment Proposal
- 1750 [Z11] Zigbee 095343, Installation Code Sample Source Code
- 1751 [Z12] Zigbee 053874, Manufacturer Code Database
- 1752 [Z13] Zigbee 115456, Master Cluster List
- 1753 [Z14] Zigbee 1602867, Zigbee 3.0 Cluster List

### 1754 1.5.2 International Standards Documents

- 1755 [I1] CIE 1931 Color Space. Commission Internationale de l'Eclairage Proceedings. Cambridge University  
1756 Press, Cambridge
- 1757 [I2] ISO 7816 International Standard for Electronic Identification Cards with Contacts (Smart Cards)

### 1758 **1.5.3 National Standards Documents**

- 1759 [N1] EN 50131 European Standards Series for Intruder Alarm Systems
- 1760 [N2] BSI British Standards, document BS EN 50523-2:2009, “Household Appliances interworking – Part 2:  
1761 Data Structures”. July 2009
- 1762 [N3] NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: CCM Mode  
1763 for Authentication and Confidentiality, May 2004
- 1764 [N4] FIPS Pub 197, Advanced Encryption Standard (AES), Processing Standards Publication 197, US  
1765 Department of Commerce/NIST Springfield, Virginia, November 26, 2001
- 1766 [N5] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information, Processing  
1767 Standards Publication 198, US Department of Commerce/NIST Springfield, Virginia, March 6, 2002

### 1768 **1.5.4 IEEE Documents**

- 1769 [E1] IEEE Standards 802, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY)  
1770 specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), IEEE, October 2003.
- 1771 [E2] IEEE 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, 1985.

### 1772 **1.5.5 ASHRAE Documents**

- 1773 [A1] ASHRAE 135-2004 standard, Data Communication Protocol for Building Automation and Control  
1774 Networks

### 1775 **1.5.6 Health Care Documents**

- 1776 [H1] ISO/IEEE 11073-20601: Health Informatics - Personal Health Device Communication - Application Profile  
1777 - Optimized Exchange Protocol - version 1.0 or later.
- 1778 [H2] ISO/IEEE P11073-10404, Health informatics – Personal health device communication – Device  
1779 specialization – Pulse oximeter.
- 1780 [H3] ISO/IEEE P11073-10407, Health informatics – Personal health device communication – Device  
1781 specialization – Blood pressure monitor.
- 1782 [H4] ISO/IEEE P11073-10408, Health informatics – Personal health device communication – Device  
1783 specialization – Thermometer.
- 1784 [H5] ISO/IEEE P11073-10415, Health informatics – Personal health device communication – Device  
1785 specialization – Weighing scale.
- 1786 [H6] ISO/IEEE P11073-10417, Health informatics – Personal health device communication – Device  
1787 specialization – Glucose meter.
- 1788 [H7] ISO/IEEE P11073-10419, Health informatics – Personal health device communication – Device  
1789 specialization – Insulin Pump
- 1790 [H8] ISO/IEEE P11073-10421, Health informatics – Personal health device communication – Device  
1791 specialization – Peak Expiratory Flow Monitor
- 1792 [H9] ISO/IEEE P11073-10441, Health informatics – Personal health device communication – Device  
1793 specialization – Cardiovascular Fitness and Activity Monitor.
- 1794 [H10] ISO/IEEE P11073-10442, Health informatics – Personal health device communication – Device  
1795 specialization – Strength Fitness Equipment.
- 1796 [H11] ISO/IEEE P11073-10471, Health informatics – Personal health device communication – Device  
1797 specialization – Independent living activity hub.

1798 [H12] ISO/IEEE P11073-10472, Health informatics – Personal health device communication – Device  
1799 specialization – Medication Monitor.

## 1800 **1.5.7 Other Documents**

1801 [O1] Standards for Efficient Cryptography: SEC 1 (working draft) ver 1.7: Elliptic Curve Cryptography,  
1802 Certicom Research, [www.secg.org](http://www.secg.org), November 13, 2006

1803 [O2] Standards for Efficient Cryptography: SEC 4 (draft) ver 1.0: Elliptic Curve Cryptography, Certicom  
1804 Research, [www.secg.org](http://www.secg.org), January 24, 2013

1805 [O3] RFC 3280: Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL)  
1806 Profile, IETF, [www.ietf.org](http://www.ietf.org), April 2002

## 1807 **1.6 Conventions**

1808 The following conventions are used in this document.

### 1809 **1.6.1 Enumerations and Reserved Values**

1810 Each undefined value or range of an enumeration, field, or identifier SHALL be considered reserved for future  
1811 revisions of this standard and SHALL not be available for implementation.

1812 Each value or range of an enumeration, field, or identifier that is available for non-standard implementation SHALL  
1813 be described as “manufacturer specific”, “vendor specific”, “ms”, or “MS”.

1814 Each value or range of an enumeration, field, or identifier that is available for other parts of this standard SHALL be  
1815 described as such.

1816 Each value or range of an enumeration, field, or identifier that is obsolete, and not available for implementation,  
1817 SHALL be described as “Obsolete”.

### 1818 **1.6.2 Reserved Bit Fields**

1819 Each full or partial data field (e.g., message data field), of any bit length, that is undefined, SHALL be considered  
1820 reserved for future revisions of this standard and SHALL not be available for implementation.

1821 Please see Section Chapter 2, Transmission and Reception, regarding rules for setting and interpreting reserved fields.

### 1822 **1.6.3 Number Format**

1823 In this specification, hexadecimal numbers are prefixed with the designation “0x” and binary numbers are prefixed  
1824 with the designation “0b”. All other numbers are assumed to be decimal unless indicated otherwise within the  
1825 associated text.

1826 Binary numbers are specified as successive groups of 4 bits, separated by a space (“ ”) character from the most  
1827 significant bit (next to the 0b prefix and left most on the page) to the least significant bit (rightmost on the page), e.g.  
1828 the binary number 0b0000 1111 represents the decimal number 15. Where individual bits are indicated (e.g. bit 3) the  
1829 bit numbers are relative to the least significant bit which is bit 0.

1830 When a bit is specified as having a value of either 0 or 1 it is specified with an “x”, e.g. “0b0000 0xxx” indicates that  
1831 the lower 3 bits can take any value but the upper 5 bits must each be set to 0.

1832

## 1.7 Testing, Validation and Certification

---

1833 The text of this document has been balloted and approved according to the procedures existing at the time of each  
1834 release. However, not all cluster specifications have undergone a complete and successful specification validation  
1835 event (SVE), which requires the testing of multiple diverse implementations to verify a common interpretation of the  
1836 text.

1837 New features may be added to new releases of this document, but only if the specification text has passed an SVE.  
1838 Old releases did not follow this rule, hence the statement above. Even a successful SVE for a cluster specification may  
1839 not test or validate all cluster specification text.

1840 This is a living document. Erroneous or ambiguous text is discovered regularly, reviewed, approved and added as  
1841 errata to this document. Testing and validation of errata may lag the release of errata text.

1842 Not all clusters in this specification are certifiable under a current certification program. A certification program  
1843 defines a testing process that may require reference implementations, test fixtures, and/or test scripts. A certification  
1844 program may also be frozen or inactive. A frozen certification program does not support new errata or new features.  
1845 If there is no active certification program or the testing process requirements are not in place to test a cluster  
1846 specification, then the cluster specification is not able to be certified.

1847 As of the writing of this section, the active and unfrozen certification programs for this specification are for Zigbee  
1848 3.0 and Smart Energy. Please see documents [Z13] and [Z14] which track the status of criteria for cluster specification  
1849 certification

## CHAPTER 2 FOUNDATION

1850

1851 The Zigbee Cluster Library is made of individual chapters such as this one. See Document Control in the Zigbee  
1852 Cluster Library for a list of all chapters and documents. References between chapters are made using a *X.Y* notation  
1853 where *X* is the chapter and *Y* is the sub-section within that chapter. References to external documents are contained in  
1854 Chapter 1 and are made using [*Rn*] notation.

### 2.1 Scope and Purpose

1855

1856 This chapter provides an entry point into the documentation for the Zigbee Cluster Library (ZCL), and specifies the  
1857 elements that are general across the entire library.

1858 The ZCL frame structure is specified along with ZCL wide commands used to manipulate attributes from all the  
1859 clusters defined throughout the ZCL. In addition, a set of data types is defined that can be used to represent attributes  
1860 and a common set of status values returned by commands throughout the ZCL.

1861 An overview is included which lists all the domains specified in the ZCL and the clusters contained therein.

### 2.2 Cluster Library Overview

1862

1863 The Zigbee Cluster Library (ZCL) is intended to act as a repository for cluster functionality and it is a working library  
1864 with regular updates as new functionality is added. A developer constructing a new application SHOULD use the ZCL  
1865 to find relevant cluster functionality that can be incorporated into the new application so as not to “re-invent the  
1866 wheel”. This also allows applications to be developed with more of an object-oriented style approach.

#### 2.2.1 Architecture and Data Model

1867

1868 Each cluster specification in this document defines an independent functional entity. Each cluster specification is  
1869 agnostic regarding functions beyond its purpose and scope, including overall requirements of the application or device.  
1870 An application cluster SHOULD have no dependencies outside its application domain. A utility cluster MAY provide  
1871 an interface to other layers (e.g. Groups cluster for group addressing).

1872 Please see [Z5] Application Architecture for more details.

##### 2.2.1.1 Cluster Identifier

1873

1874 A cluster identifier SHALL map to a single cluster specification. A cluster identifier also defines the purpose of a  
1875 cluster instance. More than one cluster identifier, each with a unique purpose, MAY map to a single more abstract  
1876 cluster specification. For example: A Concentration Measurement cluster specification MAY be quite abstract, but  
1877 have many mapped cluster identifiers each with a more concrete purpose, such as CO<sub>2</sub> measurement in air, by volume.

1878 Please see [Z5] Application Architecture for more details.

##### 2.2.1.2 Extensibility Model

1879

1880 A cluster specification MAY be derived from a base cluster specification. A derived cluster specification SHALL add  
1881 specific requirements (attributes, commands, behavior, dependencies, etc) to the base specification. A derived  
1882 specification MAY reduce optionality by limiting the optional requirements from the base specification.

1883 All new attribute and command definitions for the derived cluster SHALL be specified in the base cluster specification  
1884 as optional, to maintain, in one specification, the identifier name space and communication behavior. Other behavior  
1885 and dependencies that are specific to the derived cluster MAY also be specified in the base cluster, if it is deemed  
1886 reusable by future derived clusters.

- 1887 A derived cluster specification SHALL have the same mandatory requirements as the base cluster specification. A  
1888 derived specification MAY have mandatory requirements that are optional in the base specification.
- 1889 A derived cluster specification defines its own revision (*ClusterRevision* attribute) that is independent of the base  
1890 specification.
- 1891 Conversely, a base cluster may be defined from an original more specific cluster, which then becomes a derived  
1892 cluster.
- 1893 When considering the addition of one or more clusters to this specification, one SHALL explore the possibility of  
1894 either deriving a cluster from an existing cluster, or creating a base cluster to map or derive new and existing cluster  
1895 identifiers. This allows the reuse of approved and validated specifications and test plans.
- 1896 Please see [Z5] Application Architecture for more details.

### 1897 **2.2.1.3 Instance Model**

- 1898 If a device endpoint supports both a derived server cluster identifier and its base server cluster identifier, then both  
1899 SHALL represent a single instance and operate as a single entity. This makes it possible to deploy a new device  
1900 endpoint with both a base and derived cluster identifiers, which SHALL remain backward compatible to legacy  
1901 devices that support only the original cluster identifier.
- 1902 Cluster identifiers that are mapped to a single base cluster specification, but are defined for distinctly different  
1903 purposes, MAY exist together on a device endpoint. If there is no base cluster identifier defined, or no base cluster  
1904 identifier exists on the same endpoint, then each cluster identifier SHALL represent a separate instance.
- 1905 Please see [Z5] Application Architecture for more details.

### 1906 **2.2.1.4 Conformance Model**

- 1907 Specified behavior SHALL be Mandatory (M), Optional (O), or Deprecated (D). Mandatory behavior is usually  
1908 dependent on other factors. For example: The mandatory behavior defined in a cluster server specification, is only  
1909 mandatory, if the cluster id is discoverable as a server on the device. Attributes and commands MAY also be dependent  
1910 on the support of other optional attributes. This is true when a feature of a cluster requires a complete set of attributes  
1911 and commands.
- 1912 Deprecated attributes and commands SHALL be noted as deprecated and description text SHALL be deleted.

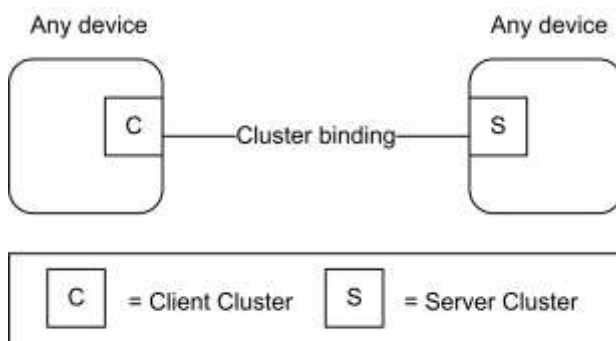
## 1913 **2.2.2 Client/Server Model**

- 1914 Throughout the ZCL, a client/server model is employed. This model is illustrated in Figure 2-1.



1915

**Figure 2-1. The ZCL Client Server Model**



1916

*Note: Device names are examples for illustration purposes only*

1917 A cluster is a related collection of commands and attributes, which together define an interface to specific  
 1918 functionality. Typically, the entity that stores the attributes of a cluster is referred to as the server of that cluster and  
 1919 an entity that affects or manipulates those attributes is referred to as the client of that cluster. However, if required,  
 1920 attributes MAY also be present on the client of a cluster.

1921 Commands that allow devices to manipulate attributes, e.g., in this document the read attribute (see 2.5.1) or write  
 1922 attribute (see 2.5.3) commands, are (typically) sent from a client device and received by the server device. Any  
 1923 response to those commands, e.g., in this document the read attribute response (see 2.5.2) or the write attribute  
 1924 response (see 2.5.5 commands), are sent from the server device and received by the client device.

1925 Conversely, the command that facilitates dynamic attribute reporting, i.e., the report attribute command (see 2.5.11)  
 1926 is (typically) sent from the server device (as typically this is where the attribute data itself is stored) and sent to the  
 1927 client device that has been bound to the server device.

1928 A type 1 cluster’s primary function is to initiate transactions from the client to the server. For example: An On/Off  
 1929 client sends commands (data) to the On/Off server. A type 2 cluster’s primary function is to initiate transactions from  
 1930 the server to the client. For example: A Temperature Measurement server reports to the Temperature Measurement  
 1931 client. Please see [Z5] Application Architecture for more details.

1932 The clusters supported by an application are identified through the simple descriptor (see [Z1]), specified on each  
 1933 active endpoint of a device. In the simple descriptor, the application input cluster list SHALL contain the list of server  
 1934 clusters supported on the device and the application output cluster list SHALL contain the list of client clusters  
 1935 supported on the device.

## 1936 **2.3 Functional Description**

1937 Global requirements for all clusters and commands are described here.

### 1938 **2.3.1 Transmission**

1939 ZCL frames are transmitted via the APS sub-layer by issuing the APSDE-DATA.request primitive.

1940 All sub-fields of ZCL frames, including individual bits, that are unspecified, or specified as reserved, SHALL be set  
 1941 to zero for transmission. This applies to all ZCL frames, including cluster-specific frames. Similarly, all reserved or  
 1942 unspecified bits of attributes of data type class Bitmap SHALL be set to zero for transmission.

### 1943 **2.3.2 Reception**

1944 ZCL frames are received via the APS sub-layer by the reception of the APSDE-DATA.indication primitive.

1945 On receipt of a command (including both general and cluster-specific commands) the device SHALL attempt to parse  
1946 and execute the command. During the parsing process for a non-manufacturer-specific command, it SHALL ignore  
1947 all reserved sub-fields of the ZCL frame, including individual reserved bits.

1948 Note that, if any of these sub-fields are not set to zero, this MAY indicate that the format or interpretation of the frame  
1949 has been updated. However, it is the responsibility of the specifier of such an updated format that it be backward  
1950 compatible, i.e., any new format will result in the same functionality as before when parsed by a device that supports  
1951 the previous version of the cluster. Any additional octets found appended to the frame SHALL also be ignored, as  
1952 these MAY be added as part of such an updated frame format.

1953 If the command is manufacturer-specific, handling of reserved sub-fields is determined by the manufacturer.

1954 If required, the device SHALL then generate a response to the command. Responses are detailed in the specification  
1955 of each command. If there is no response specified for a particular set of circumstances, (e.g., if the command has  
1956 been rejected or is not recognized, or the command has succeeded but there is no response specified to indicate  
1957 success), the Default Response command SHALL be generated, taking into account the conditions in 2.5.12.2. The  
1958 status code returned by the Default Response command SHALL be one of the status enumerations listed in Table  
1959 2-11.

### 1960 **2.3.2.1 Broadcast Endpoint**

1961 The device processing a message sent to the broadcast endpoint (0xff) SHALL:

- 1962 1. only deliver a copy of the message to the endpoints supporting the cluster indicated in the APS Header.
- 1963 2. follow the Default Response command behavior described in section 2.5.12.2 (no response for non-unicast  
1964 messages).
- 1965 3. not generate error response messages, except when required by the Default Response command behavior.

### 1966 **2.3.2.2 Broadcast Endpoint Recommendations**

1967 Broadcast Endpoint Behavior Recommendations for Avoiding Network Congestion:

1968  
1969 1. A device SHOULD NOT send a broadcast message to the broadcast endpoint where a response is expected from  
1970 every active endpoint. It is recommended to use discovery to determine the specific endpoint(s) per device and then  
1971 send individual messages that target those specific endpoints.

1972 2. A device processing a message sent to the broadcast endpoint SHOULD jitter messages that are sent in response,  
1973 especially when the nature of the message is such that it generates many responses (i.e. synchronization message).

1974  
1975 NOTE: Multicast group messages do not include an endpoint

### 1976 **2.3.3 Manufacturer Specific Extensions**

1977 Manufacturers are free to extend the standard in the following ways:

- 1978 • Add manufacturer specific clusters to a standard device endpoint.
- 1979 • Add manufacturer specific commands to a standard cluster.
- 1980 • Add manufacturer specific attributes to a standard cluster.

1981 All communications regarding manufacturer specific extensions SHALL be transmitted with the manufacturer specific  
1982 sub-field of the frame control field set to 1 and the manufacturer code included in the frame.

1983 If the manufacturer code in a command frame is not recognized, the command is not carried out.

1984 **2.3.4 Attributes**

1985 **2.3.4.1 Variables and Attributes<sup>1</sup>**

1986 A cluster variable (or variable) is a cluster data point with a defined value that is referenced in a cluster specification.  
 1987 A cluster attribute is a variable with a defined identifier, data type and access type. Optional attributes MAY be  
 1988 referenced as variables in other attribute specifications within the same cluster specification.

1989 **2.3.4.2 Dependencies on Optional Attributes**

1990 If the specification text of a cluster depends on the value of an optional attribute or variable of the same cluster, then  
 1991 the optional attribute or variable SHALL have a well-defined measured or default value. When an optional attribute  
 1992 is not supported by a device, then the default value SHALL be used for the specification text that depends on it<sup>2</sup>. This  
 1993 rule SHALL be recursive if there is a chain of dependencies.

1994 **2.3.4.3 Default Value<sup>3</sup>**

1995 If the Default Value of an attribute or variable is specified as “MS” or “ms”, then there is no default value and the  
 1996 application must return a manufacturer specific valid value that is in the operational range. An attribute or variable  
 1997 SHALL have a defined default value when:

- 1998 • an initial value is needed before the application starts
- 1999 • the value cannot be determined by the application for the instance
- 2000 • the attribute is not implemented, but there is a dependency on the attribute value

2001 If the default value is not specified in this specification, then the default value is the Illegal Value as specified in the  
 2002 data type specification (see 2.6.2), if defined. If no Illegal Value is specified, then the default value SHALL be zero.

2003 **2.3.4.4 Attribute Access**

2004 Attributes MAY support these types of access that are listed in each cluster specification’s attribute table for each  
 2005 attribute:

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

<sup>1</sup> CCB 2327 optional attributes as dependent variables in other attribute (see Dehumidification Control)

<sup>2</sup> CCB 2327 generalize dependencies in specification text

<sup>3</sup> CCB 2370 define what a default value is

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

2006 Local specific cluster commands for the cluster supporting the attribute MAY also access the attributes as defined in  
2007 each cluster specification.

### 2008 2.3.4.5 Global Attributes

2009 Cluster global attributes (see 2.6.1.3) are either mandatory or optional. All cluster instances SHALL support  
2010 mandatory global attributes.

2011 **Table 2-1. Global Attributes**

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

#### 2012 2.3.4.5.1 ClusterRevision Attribute

2013 The *ClusterRevision* global attribute is mandatory for all cluster instances, client and server, conforming to ZCL  
2014 revision 6 (ZCL6) and later ZCL revisions. This cluster attribute represents the revision of the cluster specification  
2015 that has been implemented. An implementation of a cluster specification before ZCL6 SHALL have an assumed  
2016 cluster revision of 0. The initial value for the *ClusterRevision* attribute SHALL be 1. The *ClusterRevision* attribute  
2017 SHALL be incremented and associated with each approved revision and release of a cluster specification.

2018 An implementation of a new revision of a cluster specification SHALL interoperate with an implementation of an  
2019 older revision of the cluster specification.

2020 Interoperability with a cluster MAY require reading the *ClusterRevision* attribute. For example: If a new product  
2021 application supporting revision 3 of cluster X wishes to take advantage of the new behavior that is mandated by  
2022 revision 3, then the application SHOULD read the revision of the corresponding cluster X in each remote application.  
2023 If a corresponding cluster X supports revision 3 or greater, than the behavior is supported. Conversely: Backward  
2024 compatibility MAY require that a new cluster revision read the *ClusterRevision* of a corresponding cluster to support  
2025 interoperability with legacy cluster revisions.

2026 Please see [Z5] Application Architecture for more details.

#### 2027 2.3.4.5.2 AttributeReportingStatus Attribute

2028 When reporting requires sending multiple *Report Attributes* commands, this attribute SHOULD be included in the last  
2029 attribute record, to indicate that all required attributes have been reported, or that there are still attributes pending to  
2030 be reported. The enumerated values for this attribute are outlined below:

2031 **Table 2-2. AttributeReportingStatus Enumerations**

Enumerated Value	Status
0x00	Pending

0x01	Attribute Reporting Complete
------	------------------------------

2032 **2.3.5 Persistent Data**

2033 Persistent data is persistent across a restart. A restart is a program restart (warm start) or power cycle (cold start), but  
 2034 not a factory reset.

2035 Cluster attributes that represent configuration data SHALL be persistent data unless otherwise specified. For example:  
 2036 a writeable attribute that persistently changes the behavior (or mode) of the cluster. Examples of non-configuration  
 2037 data: data that is calculated or comes from an external source, such as a sensor value, a time value, etc.

2038 Many clusters define persistent data that are not attributes. For example: The scene table that is part of a Scene cluster  
 2039 instance, or the Alarm Table in the Alarms cluster.

2040 Commissioning or configuration data that is created to allow the cluster to perform its function is persistent data. For  
 2041 example: A reporting configuration for a cluster attribute.

2042 An APS group table entry and an APS binding are both persistent data across a restart.

2043 A factory reset is a deliberate behavior to reset the above described persistent data back to its original state when the  
 2044 product left the factory.

2045 **2.4 Command Frame Formats**

2046 All commands, defined in this specification, SHALL be transmitted to the stack using the message service.

2047 The transmission order for octets and bits of all ZCL elements is as specified in section 1.2.1.3 of the Zigbee  
 2048 Specification [Z1], i.e., least significant octet and bit first.

2049 **2.4.1 General ZCL Frame Format**

2050 The ZCL frame format is composed of a ZCL header and a ZCL payload. The general ZCL frame SHALL be formatted  
 2051 as illustrated in Figure 2-2.

2052 **Figure 2-2. Format of the General ZCL Frame**

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

2053 **2.4.1.1 Frame Control Field**

2054 The frame control field is 8 bits in length and contains information defining the command type and other control flags.  
 2055 The frame control field SHALL be formatted as shown in Figure 2-3. Bits 5-7 are reserved for future use and SHALL  
 2056 be set to 0.

2057 **Figure 2-3. Format of the Frame Control Field**

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

### 2058 **2.4.1.1.1 Frame Type Sub-field**

2059 The frame type sub-field is 2 bits in length and SHALL be set to one of the non-reserved values listed in Figure 2-4.

2060 **Figure 2-4. Values of the Frame Type Sub-field**

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

2061

### 2062 **2.4.1.1.2 Manufacturer Specific Sub-field**

2063 The manufacturer specific sub-field is 1 bit in length and specifies whether this command refers to a manufacturer  
2064 specific extension. If this value is set to 1, the manufacturer code field SHALL be present in the ZCL frame. If this  
2065 value is set to 0, the manufacturer code field SHALL not be included in the ZCL frame. Manufacturer specific clusters  
2066 SHALL support global commands (Frame Type 0b00).

### 2067 **2.4.1.1.3 Direction Sub-field**

2068 The direction sub-field specifies the client/server direction for this command. If this value is set to 1, the command is  
2069 being sent from the server side of a cluster to the client side of a cluster. If this value is set to 0, the command is being  
2070 sent from the client side of a cluster to the server side of a cluster.

### 2071 **2.4.1.1.4 Disable Default Response Sub-field**

2072 The disable Default Response sub-field is 1 bit in length. If it is set to 0, the Default Response command will be  
2073 returned, under the conditions specified in 2.5.12.2. If it is set to 1, the Default Response command will only be  
2074 returned if there is an error, also under the conditions specified in 2.5.12.2.

2075 This field SHALL be set to 1, for all response frames generated as the immediate and direct effect of a previously  
2076 received frame.

## 2077 **2.4.1.2 Manufacturer Code Field**

2078 The manufacturer code field is 16 bits in length and specifies the assigned manufacturer code for proprietary  
2079 extensions. This field SHALL only be included in the ZCL frame if the manufacturer specific sub-field of the frame  
2080 control field is set to 1. Please see [Z12] Manufacturer Code Database for a list of manufacturer codes.

## 2081 **2.4.1.3 Transaction Sequence Number**

2082 The Transaction Sequence Number field is 8 bits in length and specifies an identification number for a single  
2083 transaction that includes one or more frames in both directions. Each time the first frame of a transaction is generated,  
2084 a new value SHALL be copied into the field. When a frame is generated as the specified effect on receipt of a previous  
2085 frame, then it is part of a transaction, and the Transaction Sequence Number SHALL be copied from the previously  
2086 received frame into the generated frame. This includes a frame that is generated in response to request frame.

2087 The Transaction Sequence Number field can be used by a controlling device, which MAY have issued multiple  
2088 commands, so that it can match the incoming responses to the relevant command.

2089 **2.4.1.4 Command Identifier Field**

2090 The Command Identifier field is 8 bits in length and specifies the cluster command being used. If the frame type sub-  
 2091 field of the frame control field is set to 0b00, the command identifier corresponds to one of the non-reserved values  
 2092 of Table 2-3. If the frame type sub-field of the frame control field is set to 0b01, the command identifier corresponds  
 2093 to a cluster specific command. The cluster specific command identifiers can be found in each individual document  
 2094 describing the clusters (see also 2.2.1.1).

2095 **2.4.1.5 Frame Payload Field**

2096 The frame payload field has a variable length and contains information specific to individual command types. The  
 2097 maximum payload length for a given command is limited by the stack profile in use, in conjunction with the applicable  
 2098 cluster specification and application profile. Fragmentation will be used where available.

2099 **2.5 General Command Frames**

2100 General command frames are used for manipulating attributes and other general tasks that are not specific to an  
 2101 individual cluster.

2102 The command frames defined in this document are listed in Table 2-3. Each command frame SHALL be constructed  
 2103 with the frame type sub-field of the frame control field set to 0b00.

2104 All clusters (server and client) SHALL support generation, reception and execution of the Default Response command.

2105 Except for the optional Discover Attributes Extended commands and the optional Discover Commands commands,  
 2106 each cluster (server or client) that implements attributes SHALL support reception of, execution of, and response to  
 2107 all commands to discover, read, and write these attributes. However, if no attributes with structured types are  
 2108 supported, it is not required to support the structured read and write commands.

2109 Implementation of commands to report, Configure Reporting of, and Read Reporting Configuration of attributes is  
 2110 only mandatory if the cluster has attributes whose reportability is mandatory.

2111 Generation of request commands (e.g., Read Attributes, Write Attributes, etc), is application dependent.

2112

**Table 2-3. ZCL Command Frames**

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

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

2113

2114 

## 2.5.1 Read Attributes Command

2115 

### 2.5.1.1 Read Attributes Command Frame Format

2116 The Read Attributes command frame SHALL be formatted as illustrated in Figure 2-5.

2117 **Figure 2-5. Format of the Read Attributes Command Frame**

Octets: Variable	2	2	...	2
ZCL header	Attribute identifier 1	Attribute identifier 2	...	Attribute identifier <i>n</i>

2118 

#### 2.5.1.1.1 ZCL Header Fields

2119 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2120 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Read  
 2121 Attributes defined for any cluster in the ZCL or 1 if this command is being used to read manufacturer specific  
 2122 attributes.

2123 The command identifier field SHALL be set to indicate the Read Attributes command (see Table 2-3).

2124 

#### 2.5.1.1.2 Attribute Identifier Field

2125 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be read.



2126 **2.5.1.2 When Generated**

2127 The Read Attributes command is generated when a device wishes to determine the values of one or more attributes  
 2128 located on another device. Each attribute identifier field SHALL contain the identifier of the attribute to be read.

2129 **2.5.1.3 Effect on Receipt**

2130 On receipt of this command, the device SHALL process each specified attribute identifier and generate a Read  
 2131 Attributes Response command. The Read Attributes Response command SHALL contain as many read attribute status  
 2132 records as attribute identifiers included in this command frame, subject to applicable space limitations. Each read  
 2133 attribute status record SHALL contain the corresponding attribute identifier from this command frame, a status value  
 2134 evaluated as described below, and, depending on the status value, the value of the attribute itself.

2135 For each attribute identifier included in the command frame, the device SHALL create an attribute status record as  
 2136 follows:

2137 If the attribute identifier does not correspond to an attribute that exists on this device, the device SHALL set the status  
 2138 field of the corresponding read attribute status record to UNSUPPORTED\_ATTRIBUTE and SHALL not include an  
 2139 attribute value field.

2140 If the attribute identified by the attribute identifier is supported, the device SHALL determine if the attribute status  
 2141 record carrying the attribute’s current value fits into the remaining space available in the response frame. If the status  
 2142 record does not fit, the device SHALL set the status field of the corresponding read attribute status record to  
 2143 INSUFFICIENT\_SPACE and not include the data type and value fields. Otherwise the device SHALL set the status  
 2144 field of the corresponding read attribute status record to SUCCESS and SHALL set the attribute value field to its  
 2145 current value.

2146 If the resulting attribute status record does not fit into the response frame, the device SHALL transmit the response  
 2147 frame as assembled so far and terminate this process.

2148 Otherwise, the device SHALL then move on to the next attribute identifier.

2149 **2.5.2 Read Attributes Response Command**

2150 **2.5.2.1 Read Attributes Response Command Frame Format**

2151 The Read Attributes Response command frame SHALL be formatted as illustrated in Figure 2-6.

2152 **Figure 2-6. Format of Read Attributes Response Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Read attribute status record 1	Read attribute status record 2	...	Read attribute status record <i>n</i>

2153 Each read attribute status record SHALL be formatted as illustrated in Figure 2-7.

2154 **Figure 2-7. Format of the Read Attributes Status Record Field**

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

2155 **2.5.2.1.1 ZCL Header Fields**

2156 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2157 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response  
2158 to reading attributes defined for any cluster in the ZCL or 1 if this command is being used as a response to reading  
2159 manufacturer specific attributes.

2160 The command identifier field SHALL be set to indicate the Read Attributes Response command (see Table 2-3).

2161 **2.5.2.1.2 Attribute Identifier Field**

2162 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that has been read  
2163 (or of which an element has been read). This field SHALL contain the same value that was included in the  
2164 corresponding attribute identifier field of the original Read Attributes or Read Attributes Structured command.

2165 **2.5.2.1.3 Status Field**

2166 The status field is 8 bits in length and specifies the status of the read operation on this attribute. This field SHALL be  
2167 set to SUCCESS, if the operation was successful, or an error code, as specified in 2.5.1.3, if the operation was not  
2168 successful.

2169 **2.5.2.1.4 Attribute Data Type Field**

2170 The attribute data type field SHALL contain the data type of the attribute in the same Read Attributes status record  
2171 (see 2.6.2).

2172 ). This field SHALL only be included if the associated status field contains a value of SUCCESS.

2173 **2.5.2.1.5 Attribute Value Field**

2174 The attribute value field is variable in length and SHALL contain the current value of this attribute. This field SHALL  
2175 only be included if the associated status field contains a value of SUCCESS.

2176 For an attribute or element of simple type (not array, structure, set or bag), this field has the format shown in the Table  
2177 of Data Types (see 2.6.2). For an attribute or element of type array, set or bag, this field has the format shown in Figure  
2178 2-8.

2179 **Figure 2-8. Format of the Attribute Value Field for an Array, Set or Bag**

Octets: 1	2	Variable	...	Variable
Element type	Number of elements ( <i>m</i> )	Element value 1	...	Element value <i>m</i>

2180 (NB The reason that the Element type field is before the Number of elements field is so that the latter field is in the  
2181 logical position for the zeroth element.)

2182 If the Number of elements field has the value 0xffff, this indicates that the attribute or element being read is invalid /  
2183 undefined. In this case, or if the Number of elements field has the value 0, no Element value fields are included.

2184 For an attribute or element of type structure, this field has the format shown in Figure 2-9.

2185 **Figure 2-9. Format of the Attribute Value Field for a Structure**

Octets: 2	1	Variable	...	1	Variable
Number of elements ( <i>m</i> )	Element type 1	Element value 1	...	Element type <i>m</i>	Element value <i>m</i>

2186 In both figures, the Element value subfield follows the same format as that of the attribute value field. This format is  
 2187 thus recursive to any required depth (see Selector Field for limitations).

2188 If the Number of elements field has the value 0xffff, this indicates that the attribute or element being read is invalid /  
 2189 undefined. In this case, or if the Number of elements field has the value 0, no Element type or Element value fields  
 2190 are included.

### 2191 **2.5.2.2 When Generated**

2192 The Read Attributes Response command is generated in response to a Read Attributes or Read Attributes Structured  
 2193 command. The command frame SHALL contain a read attribute status record for each attribute identifier specified in  
 2194 the original Read Attributes or Read Attributes Structured command. For each read attribute status record, the attribute  
 2195 identifier field SHALL contain the identifier specified in the original Read Attributes or Read Attributes Structured  
 2196 command. The status field SHALL contain a suitable status code, as detailed in 2.5.1.3.

2197 The attribute data type and attribute value field SHALL only be included in the read attribute status record if the  
 2198 associated status field contains a value of SUCCESS and, where present, SHALL contain the data type and current  
 2199 value, respectively, of the attribute, or element thereof, that was read.

2200 The length of this command may exceed a single frame, and thus fragmentation support may be needed to return the  
 2201 entire response. If fragmentation is not supported, only as many read attribute status records as will fit in the frame  
 2202 SHALL be returned.

### 2203 **2.5.2.3 Effect on Receipt**

2204 On receipt of this command, the originator is notified of the results of its original Read Attributes attempt and, for  
 2205 each successful request, the value of the requested attribute.

2206 If fragmentation is not supported, and some trailing attribute status records have not been returned, due to space  
 2207 limitations in the frame, the originator may issue an additional Read Attributes or Read Attributes Structured command  
 2208 to obtain their values.

## 2209 **2.5.3 Write Attributes Command**

### 2210 **2.5.3.1 Write Attributes Command Frame Format**

2211 The Write Attributes command frame SHALL be formatted as illustrated in Figure 2-10.

2212 **Figure 2-10. Format of the Write Attributes Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Write attribute record 1	Write attribute record 2	...	Write attribute record <i>n</i>

2213 Each write attribute record SHALL be formatted as illustrated in Figure 2-11.

2214 **Figure 2-11. Format of the Write Attribute Record Field**

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

### 2215 **2.5.3.1.1 ZCL Header Fields**

2216 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2217 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write  
2218 Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific  
2219 attributes.

2220 The command identifier field SHALL be set to indicate the Write Attributes command (see Table 2-3).

### 2221 **2.5.3.1.2 Attribute Identifier Field**

2222 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be written.

### 2223 **2.5.3.1.3 Attribute Data Type Field**

2224 The attribute data type field SHALL contain the data type of the attribute that is to be written.

### 2225 **2.5.3.1.4 Attribute Data Field**

2226 The attribute data field is variable in length and SHALL contain the actual value of the attribute that is to be written.

## 2227 **2.5.3.2 When Generated**

2228 The Write Attributes command is generated when a device wishes to change the values of one or more attributes  
2229 located on another device. Each write attribute record SHALL contain the identifier and the actual value of the attribute  
2230 to be written.

## 2231 **2.5.3.3 Effect on Receipt**

2232 On receipt of this command, the device SHALL attempt to process each specified write attribute record and SHALL  
2233 construct a write attribute response command (2.5.5). Each write attribute status record of the constructed command  
2234 SHALL contain the identifier from the corresponding write attribute record and a status value evaluated as described  
2235 below.

2236 For each write attribute record included in the command frame, the device SHALL make the error checks listed below,  
2237 in the order shown. If an error is detected, a corresponding write attribute status record SHALL be generated, the  
2238 status SHALL be set according to the check below, and the device SHALL move on to the next write attribute record.

- 2239 1. If the attribute is not supported on this device, the status field of the corresponding write attribute status record  
2240 SHALL be set to UNSUPPORTED\_ATTRIBUTE.
- 2241 2. If the attribute data type field is incorrect, the device SHALL set the status field of the corresponding write  
2242 attribute status record to INVALID\_DATA\_TYPE.
- 2243 3. If the attribute is designated as read only, the device SHALL set the status field of the corresponding write  
2244 attribute status record to READ\_ONLY.
- 2245 4. If the device is not currently accepting write attribute commands for the attribute, the status field of the  
2246 corresponding write attribute status record SHALL be set to NOT\_AUTHORIZED or READ\_ONLY.
- 2247 5. If the supplied value is not within the specified range of the attribute, the status field of the corresponding write  
2248 attribute status record SHALL be set to INVALID\_VALUE.
- 2249 6. If the device cannot support the supplied value, the status field of the corresponding write attribute status record  
2250 SHALL be set to INVALID\_VALUE.

2251 If the above error checks pass without generating a write attribute status record, the device SHALL write the supplied  
2252 value to the identified attribute, and SHALL move on to the next write attribute record.

2253 When all write attribute records have been processed, the device SHALL generate the constructed Write Attributes  
 2254 Response command. If there are no write attribute status records in the constructed command, indicating that all  
 2255 attributes were written successfully, a single write attribute status record SHALL be included in the command, with  
 2256 the status field set to SUCCESS and the attribute identifier field omitted.

## 2257 **2.5.4 Write Attributes Undivided Command**

2258 The Write Attributes Undivided command is generated when a device wishes to change the values of one or more  
 2259 attributes located on another device, in such a way that if any attribute cannot be written (e.g., if an attribute is not  
 2260 implemented on the device, or a value to be written is outside its valid range), no attribute values are changed.

2261 In all other respects, including generation of a Write Attributes Response command, the format and operation of the  
 2262 command is the same as that of the Write Attributes command, except that the command identifier field SHALL be  
 2263 set to indicate the Write Attributes Undivided command (see Table 2-3).

## 2264 **2.5.5 Write Attributes Response Command**

### 2265 **2.5.5.1 Write Attributes Response Command Frame Format**

2266 The Write Attributes Response command frame SHALL be formatted as illustrated in Figure 2-12.

2267 **Figure 2-12. Format of Write Attributes Response Command Frame**

<b>Octets: Variable</b>	<b>3</b>	<b>3</b>	<b>...</b>	<b>3</b>
ZCL header	Write attribute status record 1	Write attribute status record 2	...	Write attribute status record <i>n</i>

2268

2269 Each write attribute status record SHALL be formatted as illustrated in Figure 2-13.

2270 **Figure 2-13. Format of the Write Attribute Status Record Field**

<b>Octets: 1</b>	<b>2</b>
Status	Attribute identifier

#### 2271 **2.5.5.1.1 ZCL Header Fields**

2272 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2273 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response  
 2274 to writing attributes defined for any cluster in the ZCL or 1 if this command is being used as a response to writing  
 2275 manufacturer specific attributes.

2276 The command identifier field SHALL be set to indicate the Write Attributes Response command (see Table 2-3).

#### 2277 **2.5.5.1.2 Status Field**

2278 The status field is 8 bits in length and specifies the status of the write operation attempted on this attribute, as detailed  
 2279 in 2.5.3.3.

2280 Note that write attribute status records are not included for successfully written attributes, to save bandwidth. In the  
 2281 case of successful writing of all attributes, only a single write attribute status record SHALL be included in the  
 2282 command, with the status field set to SUCCESS and the attribute identifier field omitted.

**2.5.5.1.3 Attribute Identifier Field**

2284 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute on which the write  
2285 operation was attempted.

**2.5.5.2 When Generated**

2286  
2287 The Write Attributes Response command is generated in response to a Write Attributes command.

**2.5.5.3 Effect on Receipt**

2288  
2289 On receipt of this command, the device is notified of the results of its original Write Attributes command.

**2.5.6 Write Attributes No Response Command****2.5.6.1 Write Attributes No Response Command Frame Format**

2291  
2292  
2293 The Write Attributes No Response command frame SHALL be formatted as illustrated in Figure 2-14.

2294 **Figure 2-14. Write Attributes No Response Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Write attribute record 1	Write attribute record 2	...	Write attribute record <i>n</i>

2295  
2296 Each write attribute record SHALL be formatted as illustrated in Figure 2-11.

**2.5.6.1.1 ZCL Header Fields**

2298 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2299 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write  
2300 Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific  
2301 attributes.

2302 The command identifier field SHALL be set to indicate the Write Attributes No Response command (see Table 2-3).

**2.5.6.1.2 Write Attribute Records**

2303  
2304 Each write attribute record SHALL be formatted as illustrated in Figure 2-11. Its fields have the same meaning and  
2305 contents as the corresponding fields of the Write Attributes command.

**2.5.6.2 When Generated**

2306  
2307 The Write Attributes No Response command is generated when a device wishes to change the value of one or more  
2308 attributes located on another device but does not require a response. Each write attribute record SHALL contain the  
2309 identifier and the actual value of the attribute to be written.

**2.5.6.3 Effect on Receipt**

2310  
2311 There SHALL NOT be any response, error response, or Default Response command, to this command.

2312 On receipt of this command, the device SHALL attempt to process each specified write attribute record.

2313 For each write attribute record included in the command frame, the device SHALL first check that it corresponds to  
 2314 an attribute that is implemented on this device. If it does not, the device SHALL ignore the attribute and move on to  
 2315 the next write attribute record.

2316 If the attribute identified by the attribute identifier is supported, the device SHALL check whether the attribute is  
 2317 writable. If the attribute is designated as read only, the device SHALL ignore the attribute and move on to the next  
 2318 write attribute record.

2319 If the attribute is writable, the device SHALL check that the supplied value in the attribute data field is within the  
 2320 specified range of the attribute. If the supplied value does not fall within the specified range of the attribute, the device  
 2321 SHALL ignore the attribute and move on to the next write attribute record.

2322 If the value supplied in the attribute data field is within the specified range of the attribute, the device SHALL write  
 2323 the supplied value to the identified attribute and move on to the next write attribute record.

## 2.5.7 Configure Reporting Command

2324 The Configure Reporting command is used to configure the reporting mechanism for one or more of the attributes of  
 2325 a cluster.  
 2326

2327 The individual cluster definitions specify which attributes SHALL be available to this reporting mechanism, however  
 2328 specific implementations of a cluster may make additional attributes available.

2329 Note that attributes with data types of array, structure, set or bag cannot be reported.

### 2.5.7.1 Configure Reporting Command Frame Format

2330 The Configure Reporting command frame SHALL be formatted as illustrated in Figure 2-15.

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

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Attribute reporting configuration record 1	Attribute reporting configuration record 2	...	Attribute reporting configuration record <i>n</i>

2333 There SHALL be one attribute reporting configuration record for each attribute to be configured. Each such record  
 2334 SHALL be formatted as illustrated in Figure 2-16.

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

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

#### 2.5.7.1.1 ZCL Header Fields

2336 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2337 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to configure  
 2338 attribute reports defined for any cluster in the ZCL or 1 if this command is being used to configure attribute reports  
 2339 for manufacturer specific attributes.  
 2340

2341 The command identifier field SHALL be set to indicate the report configuration command (see Table 2-3).

2342 **2.5.7.1.2 Direction Field**

2343 The direction field specifies whether values of the attribute are to be reported, or whether reports of the attribute are  
2344 to be received.

2345 If this value is set to 0x00, then the attribute data type field, the minimum reporting interval field, the maximum  
2346 reporting interval field and the reportable change field are included in the payload, and the timeout period field is  
2347 omitted. The record is sent to a cluster server (or client) to configure how it sends reports to a client (or server) of the  
2348 same cluster.

2349 If this value is set to 0x01, then the timeout period field is included in the payload, and the attribute data type field,  
2350 the minimum reporting interval field, the maximum reporting interval field and the reportable change field are omitted.  
2351 The record is sent to a cluster client (or server) to configure how it SHOULD expect reports from a server (or client)  
2352 of the same cluster.

2353 All other values of this field are reserved.

2354 **Table 2-4. Destination of Reporting Based on Direction Field**

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

2355 **2.5.7.1.3 Attribute Identifier Field**

2356 If the direction field is 0x00, this field contains the identifier of the attribute that is to be reported. If instead the  
2357 direction field is 0x01, the device SHALL expect reports of values of this attribute.

2358 **2.5.7.1.4 Attribute Data Type Field**

2359 The Attribute data type field contains the data type of the attribute that is to be reported.

2360 **2.5.7.1.5 Minimum Reporting Interval Field**

2361 The minimum reporting interval field is 16 bits in length and SHALL contain the minimum interval, in seconds,  
2362 between issuing reports of the specified attribute.

2363 If this value is set to 0x0000, then there is no minimum limit, unless one is imposed by the specification of the cluster  
2364 using this reporting mechanism or by the application.

2365 **2.5.7.1.6 Maximum Reporting Interval Field**

2366 The maximum reporting interval field is 16 bits in length and SHALL contain the maximum interval, in seconds,  
2367 between issuing reports of the specified attribute.

2368 If this value is set to 0xffff, then the device SHALL not issue reports for the specified attribute, and the configuration  
2369 information for that attribute need not be maintained. (**Note:** in an implementation using dynamic memory allocation,  
2370 the memory space for that information may then be reclaimed).



2371 If this value is set to 0x0000, and the minimum reporting interval field does not equal 0xffff there SHALL be no  
2372 periodic reporting, but change based reporting SHALL still be operational.<sup>4</sup>

2373 If this value is set to 0x0000 and the Minimum Reporting Interval Field equals 0xffff, then the device SHALL revert  
2374 to its default reporting configuration. The reportable change field, if present, SHALL be set to zero.

2375

### 2376 **2.5.7.1.7 Reportable Change Field**

2377 The reportable change field SHALL contain the minimum change to the attribute that will result in a report being  
2378 issued. This field is of variable length. For attributes with 'analog' data type (see 2.6.2), the field has the same data  
2379 type as the attribute. The sign (if any) of the reportable change field is ignored.

2380 For attributes of 'discrete' data type (see 2.6.2), this field is omitted.

2381 If the Maximum Reporting Interval Field is set to 0xffff (terminate reporting configuration)<sup>5</sup>, or the Maximum  
2382 Reporting Interval Field is set to 0x0000 and the Minimum Reporting Interval Field equals 0xffff, indicating a (default  
2383 reporting configuration) then if this field is present, it SHALL be set to zero upon transmission and ignored upon  
2384 reception.

### 2385 **2.5.7.1.8 Timeout Period Field**

2386 The timeout period field is 16 bits in length and SHALL contain the maximum expected time, in seconds, between  
2387 received reports for the attribute specified in the attribute identifier field. If more time than this elapses between  
2388 reports, this may be an indication that there is a problem with reporting.

2389 If this value is set to 0x0000, reports of the attribute are not subject to timeout.

2390 Note that, for a server/client connection to work properly using automatic reporting, the timeout value set for attribute  
2391 reports to be received by the client (or server) cluster must be set somewhat higher than the maximum reporting  
2392 interval set for the attribute on the server (or client) cluster.

### 2393 **2.5.7.2 When Generated**

2394 The report configuration command is generated when a device wishes to configure a device to automatically report  
2395 the values of one or more of its attributes, or to receive such reports.

### 2396 **2.5.7.3 Effect on Receipt**

2397 On receipt of this command, the device SHALL attempt to process each attribute reporting configuration record and  
2398 SHALL construct a Configure Reporting Response command. Each attribute status record of the constructed  
2399 command SHALL contain an identifier from an attribute reporting configuration record and a status value evaluated  
2400 as described below.

2401 If the direction field is 0x00, indicating that the reporting intervals and reportable change are being configured, then

- 2402 • If the attribute specified in the attribute identifier field is not implemented on this device or if the attribute type  
2403 is set to array, structure, set or bag, the device SHALL construct an attribute status record with the status field  
2404 set to UNSUPPORTED\_ATTRIBUTE.
- 2405 • Else, if the attribute identifier in this field cannot be reported (because it is not in the list of mandatory  
2406 reportable attributes in the relevant cluster specification, and support has also not been implemented as a  
2407 manufacturer option), the device SHALL construct an attribute status record with the status field set to  
2408 UNREPORTABLE\_ATTRIBUTE.

---

<sup>4</sup> Moved from Report Attributes command description

<sup>5</sup> CCB 2338 clarify the value of this field as dependent on the Maximum Reporting Interval field

- 2409 • Else, if the attribute data type field is incorrect, the device SHALL construct an attribute status record with the  
2410 status field set to INVALID\_DATA\_TYPE.
- 2411 • Else, if the minimum reporting interval field is less than any minimum set by the relevant cluster specification  
2412 or application, or the value of the maximum reporting interval field is non-zero and is less than that of the  
2413 minimum reporting interval field, the device SHALL construct an attribute status record with the status field set  
2414 to INVALID\_VALUE.
- 2415 • Else, if the value of the minimum or maximum reporting interval field is not supported by the product, the  
2416 device SHALL construct an attribute status record with the status field set to INVALID\_VALUE.
- 2417 • Else the device SHALL set the minimum and maximum reporting intervals and the reportable change for the  
2418 attribute to the values contained in the corresponding fields.
- 2419 Else the direction field is 0x01, indicating that the timeout period is being configured, then
- 2420 If reports of values of the attribute identifier specified in the attribute identifier field cannot be received (because it is  
2421 not in the list of mandatory reportable attributes in the relevant cluster specification, and support has also not been  
2422 implemented as a manufacturer option), or the timeout feature is not supported, the device SHALL construct an  
2423 attribute status record with the status field set to UNSUPPORTED\_ATTRIBUTE.
- 2424 Else the device SHALL set the timeout value for the attribute identifier specified in the attribute identifier field to the  
2425 value of the timeout period field. Note that the action to be taken by the device if the timeout period is exceeded is  
2426 cluster and device dependent, including optionally taking no action.
- 2427 When all attribute reporting configuration records have been processed, the device SHALL generate the constructed  
2428 Configure Reporting Response command. If there are no attribute status records in the constructed command,  
2429 indicating that all attributes were configured successfully, a single attribute status record SHALL be included in the  
2430 command, with the status field set to SUCCESS and the direction and attribute identifier fields omitted.
- 2431 The device SHALL then proceed to generate or receive attribute reports according the configuration just set up, by  
2432 means of the Report Attributes command (see 2.5.11.2.1 through 2.5.11.2.4). See Table 2-4 to determine the  
2433 destination of the Report Attributes command.

## 2434 2.5.8 Configure Reporting Response Command

2435 The Configure Reporting Response command is used to respond to a Configure Reporting command.

### 2436 2.5.8.1 Configure Reporting Response Command Frame 2437 Format

2438 The Configure Reporting Response command frame SHALL be formatted as illustrated in Figure 2-17.

2439 **Figure 2-17. Format of the Configure Reporting Response Command Frame**

Octets: Variable	4	4	...	4
ZCL header	Attribute status record 1	Attribute status record 2	...	Attribute status record <i>n</i>

2440 Each attribute status record SHALL be formatted as illustrated in Figure 2-18.

2441 **Figure 2-18. Format of the Attribute Status Record Field**

Octets: 1	1	2
Status	Direction	Attribute identifier

2442 **2.5.8.1.1 ZCL Header Fields**

2443 The frame control field is specified as follows. The frame type sub-field SHALL be set to indicate a global command  
 2444 (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used as a response to  
 2445 configuring attribute reports defined for any cluster in the ZCL or 1 if this command is being used as a response to  
 2446 configuring attribute reports for manufacturer specific attributes.  
 2447 The command identifier field SHALL be set to indicate the report configuration response command (see Table 2-3).

2448 **2.5.8.1.2 Direction Field**

2449 The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are  
 2450 received (0x01).  
 2451 All other values of this field are reserved.

2452 **2.5.8.1.3 Status Field**

2453 The status field specifies the status of the Configure Reporting operation attempted on this attribute, as detailed in  
 2454 2.5.7.3.  
 2455 Note that attribute status records are not included for successfully configured attributes, to save bandwidth. In the case  
 2456 of successful configuration of all attributes, only a single attribute status record SHALL be included in the command,  
 2457 with the status field set to SUCCESS and the direction and attribute identifier fields omitted.

2458 **2.5.8.2 When Generated**

2459 The Configure Reporting Response command is generated in response to a Configure Reporting command.

2460 **2.5.8.3 Effect on Receipt**

2461 On receipt of this command, the device is notified of the success (or otherwise) of its original Configure Reporting  
 2462 command, for each attribute.

2463 **2.5.9 Read Reporting Configuration Command**

2464 The Read Reporting Configuration command is used to read the configuration details of the reporting mechanism for  
 2465 one or more of the attributes of a cluster.

2466 **2.5.9.1 Read Reporting Configuration Command Frame  
 2467 Format**

2468 The Read Reporting Configuration command frame SHALL be formatted as illustrated in Figure 2-19.

2469 **Figure 2-19. Read Reporting Configuration Command Frame**

<b>Octets: Variable</b>	<b>3</b>	<b>3</b>	<b>...</b>	<b>3</b>
ZCL header	Attribute record 1	Attribute record 2	...	Attribute record <i>n</i>

2470

2471 Each attribute record SHALL be formatted as illustrated in Figure 2-20.

2472 **Figure 2-20. Format of the Attribute Status Record Field**

Octets: 1	2
Direction	Attribute identifier

### 2473 **2.5.9.1.1 ZCL Header Fields**

2474 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2475 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to read the  
2476 reporting configuration of attributes defined for any cluster in the ZCL or 1 if this command is being used to read the  
2477 reporting configuration of manufacturer specific attributes.

2478 The command identifier field SHALL be set to indicate the Read Reporting Configuration command (see Table 2-3).

### 2479 **2.5.9.1.2 Direction Field**

2480 The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are  
2481 received (0x01).

2482 All other values of this field are reserved.

### 2483 **2.5.9.1.3 Attribute Identifier Field**

2484 The attribute identifier field SHALL contain the identifier of the attribute whose reporting configuration details are to  
2485 be read.

## 2486 **2.5.9.2 Effect on Receipt**

2487 On receipt of this command, a device SHALL generate a Read Reporting Configuration Response command  
2488 containing the details of the reporting configuration for each of the attributes specified in the command (see 2.5.10).

## 2489 **2.5.10 Read Reporting Configuration Response Command**

2490 The Read Reporting Configuration Response command is used to respond to a Read Reporting Configuration  
2491 command.

### 2492 **2.5.10.1 Read Reporting Configuration Response Command 2493 Frame Format**

2494 The Read Reporting Configuration Response command frame SHALL be formatted as illustrated in Figure 2-21.

2495 **Figure 2-21. Format of the Read Reporting Configuration Response Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Attribute reporting configuration record 1	Attribute reporting configuration record 2	...	Attribute reporting configuration record <i>n</i>

2496

2497 There SHALL be one attribute reporting configuration record for each attribute record of the received Read Reporting  
 2498 Configuration command. Each such record SHALL be formatted as illustrated in Figure 2-22.

2499 **Figure 2-22. Attribute Reporting Configuration Record Field**

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

2500 **2.5.10.1.1 ZCL Header Fields**

2501 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2502 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to for  
 2503 attributes specified in the ZCL or 1 if this command is being used for manufacturer specific attributes.

2504 The command identifier field SHALL be set to indicate the Read Reporting Configuration Response command (see  
 2505 Table 2-3).

2506 **2.5.10.1.2 Status Field**

2507 If the attribute is not implemented on the sender or receiver of the command, whichever is relevant (depending on  
 2508 direction), this field SHALL be set to UNSUPPORTED\_ATTRIBUTE. If the attribute is supported, but is not capable  
 2509 of being reported, this field SHALL be set to UNREPORTABLE\_ATTRIBUTE. If the attribute is supported and  
 2510 reportable, but there is no report configuration, this field SHALL be set to NOT\_FOUND. Otherwise, this field  
 2511 SHALL be set to SUCCESS.

2512 If the status field is not set to SUCCESS, all fields except the direction and attribute identifier fields SHALL be  
 2513 omitted.

2514 **2.5.10.1.3 Direction Field**

2515 The direction field specifies whether values of the attribute are reported (0x00), or whether reports of the attribute are  
 2516 received (0x01).

2517 If this value is set to 0x00, then the attribute data type field, the minimum reporting interval field, the maximum  
 2518 reporting interval field and the reportable change field are included in the payload, and the timeout period field is  
 2519 omitted. If this value is set to 0x01, then the timeout period field is included in the payload, and the attribute data type  
 2520 field, the minimum reporting interval field, the maximum reporting interval field and the reportable change field are  
 2521 omitted.

2522 All other values of this field are reserved.

2523 **2.5.10.1.4 Attribute Identifier Field**

2524 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that the reporting  
 2525 configuration details apply to.

2526 **2.5.10.1.5 Minimum Reporting Interval Field**

2527 The minimum reporting interval field is 16 bits in length and SHALL contain the minimum interval, in seconds,  
 2528 between issuing reports for the attribute specified in the attribute identifier field. If the minimum reporting interval  
 2529 has not been configured, this field SHALL contain the value 0xffff.

2530 **2.5.10.1.6 Maximum Reporting Interval Field**

2531 The maximum reporting interval field is 16 bits in length and SHALL contain the maximum interval, in seconds,  
2532 between issuing reports for the attribute specified in the attribute identifier field. If the maximum reporting interval  
2533 has not been configured, this field SHALL contain the value 0xffff.

2534 **2.5.10.1.7 Reportable Change Field**

2535 The reportable change field SHALL contain the minimum change to the attribute that will result in a report being  
2536 issued. For attributes with 'analog' data type (see 2.6.2), the field has the same data type as the attribute. If the reportable  
2537 change has not been configured, this field SHALL contain the invalid value for the relevant data type.

2538 For attributes of 'discrete' data (see 2.6.2), this field is omitted.

2539 **2.5.10.1.8 Timeout Period Field**

2540 The timeout period field is 16 bits in length and SHALL contain the maximum expected time, in seconds, between  
2541 received reports for the attribute specified in the attribute identifier field. If the timeout period has not been configured,  
2542 this field SHALL contain the value 0xffff.

2543 **2.5.10.2 When Generated**

2544 The Read Reporting Configuration Response command is generated in response to a Read Reporting Configuration  
2545 command. Only as many attribute reporting configuration records as will fit in the frame SHALL be returned.

2546 **2.5.10.3 Effect on Receipt**

2547 On receipt of this command, the originator is notified of the results of its original Read Reporting Configuration  
2548 command.

2549 If some trailing attribute reporting configuration records have not been returned, due to space limitations in the frame,  
2550 the originator may issue a further Read Reporting Configuration command to obtain their values.

2551 **2.5.11 Report Attributes Command**

2552 The Report Attributes command is used by a device to report the values of one or more of its attributes to another  
2553 device. Individual clusters, defined elsewhere in the ZCL, define which attributes are to be reported and at what  
2554 interval. See 2.5.7 to determine the destination of the Report Attributes command.

2555 **2.5.11.1 Report Attributes Command Frame Format**

2556 The Report Attributes command frame SHALL be formatted as illustrated in Figure 2-23.

2557 **Figure 2-23. Format of the Report Attributes Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Attribute report 1	Attribute report 2	...	Attribute report <i>n</i>

2558 Each attribute report field SHALL be formatted as illustrated in Figure 2-24.

2559 **Figure 2-24. Format of the Attribute Report Fields**

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

2560 **2.5.11.1.1 ZCL Header Fields**

2561 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2562 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Report  
 2563 Attributes defined for any cluster in the ZCL or 1 if this command is being used to report manufacturer specific  
 2564 attributes.

2565 The command identifier field SHALL be set to indicate the Report Attributes command (see Table 2-3).

2566 **2.5.11.1.2 Attribute Identifier Field**

2567 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is being reported.  
 2568 When reporting requires sending multiple *Report Attributes* commands see 2.3.4.5.2.

2569 **2.5.11.1.3 Attribute Data Type Field**

2570 The attribute data type field contains the data type of the attribute that is being reported.

2571 **2.5.11.1.4 Attribute Data Field**

2572 The attribute data field is variable in length and SHALL contain the actual value of the attribute being reported.

2573 **2.5.11.2 When Generated**

2574 The Report Attributes command is generated when a device has been configured to report the values of one or more  
 2575 of its attributes to another device., and when the conditions that have been configured are satisfied. These conditions  
 2576 are detailed in the following sections.

2577 A Report Attributes command may also be configured locally on a device at any time. Except for the source, a locally  
 2578 created report configuration SHALL be no different than a configuration received externally. A locally created report  
 2579 configuration SHALL support the same services as a configuration received externally.

2580 If the destination of the Report Attributes Command cannot be determined, then the command SHALL not be  
 2581 generated. See 2.5.7 to determine the destination of the Report Attributes command.

2582 **2.5.11.2.1 Periodic Reporting**

2583 A report SHALL be generated when the time that has elapsed since the previous report of the same attribute is equal  
 2584 to the Maximum Reporting Interval for that attribute (see 2.5.7.1.6). The time of the first report after configuration is  
 2585 not specified.

2586 **2.5.11.2.2 Changes to 'Discrete' Attributes**

2587 If the attribute has a 'discrete' data type, a report SHALL be generated when the attribute undergoes any change of  
 2588 value. Discrete types are general data types (which are often used as sets of bit fields), logical types, bitmap types,  
 2589 enumerations, strings, identifiers, IEEE address and security key (see 2.6.2).

2590 Reporting is subject to the Minimum Reporting Interval for that attribute (see 2.5.7.1.5). After a report, no further  
2591 reports are sent during this interval.

### 2592 **2.5.11.2.3 Changes to 'Analog' Attributes**

2593 If the attribute has an 'analog' data type, a report SHALL be generated when the attribute undergoes a change of value,  
2594 in a positive or negative direction, equal to or greater than the Reportable Change for that attribute (see 2.5.7.1.7). The  
2595 change is measured from the value of the attribute when the Reportable Change is configured, and thereafter from the  
2596 previously reported value of the attribute.

2597 Analog types are signed and unsigned integer types, floating point types and time types (see 2.6.2).

2598 Reporting is subject to the Minimum Reporting Interval for that attribute (see 2.5.7.1.5). After a report, no further  
2599 reports are sent during this interval.

### 2600 **2.5.11.2.4 Cluster Specific Conditions**

2601 The specification for a cluster may add additional conditions for specific attributes of that cluster.

### 2602 **2.5.11.2.5 Consolidation of Attribute Reporting**

2603 To reduce the resources (such as the number of timers) required for attribute reporting, a device may adapt the timing  
2604 of reports by relaxing the configured minimum and maximum periods as described below. By employing these  
2605 techniques, a device may limit the number of timers required to any manufacturer specific value, including use of only  
2606 a single timer, though at the cost of some side effects, such as increased network traffic in some cases.

2607 In consolidating timers, several principles apply:

- 2608 1. The maximum reporting interval of an attribute may be reduced, as it SHOULD not normally cause a problem  
2609 to devices to receive reports more frequently than expected – typical reporting intervals are seconds to minutes.  
2610 It may not be increased, as this may be incompatible with any timeout period set.
- 2611 2. The minimum reporting interval of an attribute may also be reduced. However, it may not be increased, as an  
2612 application may be relying on receiving reports of changes to an attribute within a given delay time. Minimum  
2613 values are generally used to reduce network traffic, but this is less important than ensuring that the application  
2614 timing needs are satisfied.
- 2615 3. From (1), when consolidating the maximum reporting periods of two or more attributes together, the  
2616 consolidated reporting period SHALL be equal to the lowest of the configured maximum intervals of the  
2617 attributes to be reported.
- 2618 4. Similarly, from (2), when consolidating the minimum reporting periods of two or more attributes together, the  
2619 consolidated reporting period SHALL be equal to the lowest of the configured minimum intervals of the  
2620 attributes to be reported.

2621 As a first step, timers for attributes on the same cluster may be consolidated. Such adaptations SHOULD aim to send  
2622 attribute reports for different attributes of the same cluster at the same time, so that they can be consolidated into fewer  
2623 attribute reports, thus reducing network traffic.

2624 To reduce the number of timers further, timers may be consolidated across clusters and endpoints if needed.

2625 (Note that it is not generally possible to consolidate timeout values (see 2.5.7.1.8) of received attribute reports.)

### 2626 **2.5.11.3 Effect on Receipt**

2627 On receipt of this command, a device is notified of the latest values of one or more of the attributes of another device.



2628 **2.5.12 Default Response Command**

2629 **2.5.12.1 Default Response Command Frame Format**

2630 The Default Response command frame SHALL be formatted as illustrated in Figure 2-25.

2631 **Figure 2-25. Format of the Default Response Command Frame**

<b>Octets: Variable</b>	<b>1</b>	<b>1</b>
ZCL header	Command identifier	Status code

2632 **2.5.12.1.1 ZCL Header Fields**

2633 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2634 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being sent in response  
 2635 to a command defined for any cluster in the ZCL or 1 if this command is being sent in response to a manufacturer  
 2636 specific command.

2637 The command identifier sub-field SHALL be set to indicate the Default Response command (see Table 2-3).

2638 **2.5.12.1.2 Command Identifier Field**

2639 The command identifier field is 8 bits in length and specifies the identifier of the received command to which this  
 2640 command is a response.

2641 **2.5.12.1.3 Status Code Field**

2642 The status code field is 8 bits in length and specifies either SUCCESS or the nature of the error that was detected in  
 2643 the received command. It SHALL be one of the status enumerations listed in Table 2-11.

2644 **2.5.12.2 When Generated**

2645 The Default Response command SHALL be generated when all 4 of these criteria are met:

- 2646 1. A device receives a unicast command that is not a Default Response command.
- 2647 2. No other command is sent in response to the received command, using the same Transaction sequence number  
 2648 as the received command.
- 2649 3. The Disable Default Response bit of its Frame control field is set to 0 (see 2.4.1.1.4) or when an error results.
- 2650 4. The “Effect on Receipt” clause for the received command does not override the behavior of when a Default  
 2651 Response command is sent.

2652 If a device receives a command in error through a broadcast or multicast transmission, the command SHALL be  
 2653 discarded and the Default Response command SHALL not be generated.

2654 If the identifier of the received command is not supported on the device, it SHALL set the command identifier field  
 2655 to the value of the identifier of the command received in error. The status code field SHALL be set to the either:  
 2656 UNSUP\_CLUSTER\_COMMAND,  
 2657 UNSUP\_GENERAL\_COMMAND,  
 2658 UNSUP\_MANUF\_CLUSTER\_COMMAND or UNSUP\_MANUF\_GENERAL\_COMMAND, as appropriate.

2659 If the device receives a unicast cluster command to a particular endpoint, and the cluster does not exist on the endpoint,  
 2660 the status code field SHALL be set to UNSUPPORTED\_CLUSTER. Receiving devices SHOULD accept other error  
 2661 status codes, such as FAILURE, from devices certified before ZCL revision 6.

2662 The Default Response command SHALL be generated in response to reception of all commands, including response  
2663 commands (such as the Write Attributes Response command), under the conditions specified above. However, the  
2664 Default Response command SHALL not be generated in response to reception of another Default Response command.

### 2665 **2.5.12.3 Effect on Receipt**

2666 On receipt of this command, the device is notified of the success or otherwise of the generated command with the  
2667 same transaction sequence number (see 2.4.1.3).

## 2668 **2.5.13 Discover Attributes Command**

### 2669 **2.5.13.1 Discover Attributes Command Frame Format**

2670 The Discover Attributes command frame SHALL be formatted as illustrated in Figure 2-26.

2671 **Figure 2-26. Format of the Discover Attributes Command Frame**

<b>Octets: Variable</b>	<b>2</b>	<b>1</b>
ZCL header	Start attribute identifier	Maximum attribute identifiers

#### 2672 **2.5.13.1.1 ZCL Header Fields**

2673 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2674 command (0b00). The manufacturer specific sub-field SHALL be set to 0 to discover standard attributes in a cluster  
2675 or 1 to discover manufacturer specific attributes in either a standard or a manufacturer specific cluster.

2676 The command identifier field SHALL be set to indicate the Discover Attributes command (see Table 2-3).

#### 2677 **2.5.13.1.2 Start Attribute Identifier Field**

2678 The start attribute identifier field is 16 bits in length and specifies the value of the identifier at which to begin the  
2679 attribute discovery.

#### 2680 **2.5.13.1.3 Maximum Attribute Identifiers Field**

2681 The maximum attribute identifiers field is 8 bits in length and specifies the maximum number of attribute identifiers  
2682 that are to be returned in the resulting Discover Attributes Response command.

### 2683 **2.5.13.2 When Generated**

2684 The Discover Attributes command is generated when a remote device wishes to discover the identifiers and types of  
2685 the attributes on a device which are supported within the cluster to which this command is directed.

### 2686 **2.5.13.3 Effect on Receipt**

2687 On receipt of this command, the device SHALL construct an ordered list of attribute information records, each  
2688 containing a discovered attribute identifier and its data type, in ascending order of attribute identifiers. This list SHALL  
2689 start with the first attribute that has an identifier that is equal to or greater than the identifier specified in the start  
2690 attribute **identifier field**. **The number of attribute identifiers included in the list SHALL not exceed that specified**  
2691 **in the maximum attribute identifiers field.**

2692 The device SHALL then generate a Discover Attributes Response command containing the discovered  
 2693 attributes and their types, and SHALL return it to the originator of the Discover Attributes command.

## 2694 2.5.14 Discover Attributes Response Command

### 2695 2.5.14.1 Discover Attributes Response Command Frame 2696 Format

2697 The Discover Attributes Response command frame SHALL be formatted as illustrated in Figure 2-27.

2698 **Figure 2-27. Discover Attributes Response Command Frame**

<b>Octets: Variable</b>	<b>1</b>	<b>3</b>	<b>3</b>	<b>...</b>	<b>3</b>
ZCL header	Discovery complete	Attribute information 1	Attribute information 2	...	Attribute information <i>n</i>

2699 Each attribute information field SHALL be formatted as illustrated in Figure 2-28.  
 2700

2701 **Figure 2-28. Format of the Attribute Report Fields**

<b>Octets: 2</b>	<b>1</b>
Attribute identifier	Attribute data type

#### 2702 2.5.14.1.1 ZCL Header Fields

2703 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2704 command (0b00). The manufacturer specific sub-field SHALL be set to the same value included in the original  
 2705 Discover Attributes command.

2706 The command identifier field SHALL be set to indicate the Discover Attributes Response command (see Table 2-3).

#### 2707 2.5.14.1.2 Discovery Complete Field

2708 The discovery complete field is a Boolean field. A value of 0 indicates that there are more attributes to be discovered  
 2709 that have an attribute identifier value greater than the last attribute identifier in the last attribute information field. A  
 2710 value of 1 indicates that there are no more attributes to be discovered.

#### 2711 2.5.14.1.3 Attribute Identifier Field

2712 The attribute identifier field SHALL contain the identifier of a discovered attribute. Attributes SHALL be included in  
 2713 ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute identifier  
 2714 field of the received Discover Attributes command.

#### 2715 2.5.14.1.4 Attribute Data Type Field

2716 The attribute data type field SHALL contain the data type of the attribute in the same attribute report field (see 2.6.2).

2717 **2.5.14.2 When Generated**

2718 The Discover Attributes Response command is generated in response to a Discover Attributes command.

2719 **2.5.14.3 Effect on Receipt**

2720 On receipt of this command, the device is notified of the results of its attribute discovery request.

2721 Following the receipt of this command, if the discovery complete field indicates that there are more attributes to be  
 2722 discovered, the device may choose to send subsequent discover attribute request commands to obtain the rest of the  
 2723 attribute identifiers. In this case, the start attribute identifier specified in the next attribute discovery request command  
 2724 SHOULD be set equal to one plus the last attribute identifier received in the Discover Attributes Response command.

2725 **2.5.15 Read Attributes Structured Command**2726 **2.5.15.1 Read Attributes Structured Command Frame Format**

2727 The Read Attributes Structured command frame SHALL be formatted as illustrated in Figure 2-29.

2728 **Figure 2-29. Format of Read Attributes Structured Command Frame**

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

2729 **2.5.15.1.1 ZCL Header Fields**

2730 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2731 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Read  
 2732 Attributes defined for any cluster in the ZCL or 1 if this command is being used to read manufacturer specific  
 2733 attributes.

2734 The command identifier field SHALL be set to indicate the Read Attributes Structured command (see Table 2-3).

2735 **2.5.15.1.2 Attribute Identifier Field**

2736 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be read.

2737 **2.5.15.1.3 Selector Field**

2738 Each attribute identifier field is followed by a selector field, which specifies whether the whole of the attribute value  
 2739 is to be read, or only an individual element of it. An individual element may only be read from attributes with types  
 2740 of Array or Structure.

2741 The Selector field SHALL be formatted as illustrated in Figure 2-30.

2742 **Figure 2-30. Format of the Selector Field**

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

2743

2744 The Indicator subfield indicates the number of index fields that follow it. This number is limited to the range 0 - 15.  
 2745 It may be further limited by an application. All other values of this field are reserved.

2746 If this subfield is 0, there are no index fields, and the whole of the attribute value is to be read. For attributes of type  
 2747 other than array or structure, this subfield SHALL have the value 0.

2748 If this subfield is 1 or greater, the index fields indicate which element is to be read, nested to a depth of m. For example,  
 2749 if the attribute is an array of arrays (or structures), then if m = 2, index 1 = 5 and index 2 = 3, the third element of the  
 2750 fifth element of the attribute will be read.

2751 Note that elements are numbered from 1 upwards for both arrays and structures. The zeroth element of an array or  
 2752 structure is readable, always has type 16 bit unsigned integer, and returns the number of elements contained in the  
 2753 array or structure.

2754 **2.5.15.2 When Generated**

2755 The Read Attributes command is generated when a device wishes to determine the values of one or more attributes,  
 2756 or elements of attributes, located on another device. Each attribute identifier field SHALL contain the identifier of the  
 2757 attribute to be read.

2758 **2.5.15.3 Effect on Receipt**

2759 On receipt of this command, the device SHALL process each specified attribute identifier and associated selector, and  
 2760 SHALL generate a Read Attributes Response command. The Read Attributes Response command SHALL contain as  
 2761 many read attribute status records as there are attribute identifiers included in this command frame. Each read attribute  
 2762 status record SHALL contain the corresponding attribute identifier from this command frame, a status value evaluated  
 2763 as described below, and, depending on the status value, the value of the attribute (or attribute element) itself.

2764 For each attribute identifier included in the command frame, the device SHALL first check that it corresponds to an  
 2765 attribute that exists on this device, and that its associated selector field correctly indicates either the whole of the  
 2766 attribute or an element of the attribute. If it does not, the device SHALL set the status field of the corresponding read  
 2767 attribute status record to either UNSUPPORTED\_ATTRIBUTE or INVALID\_SELECTOR as appropriate, and  
 2768 SHALL not include an attribute value field. The device SHALL then move on to the next attribute identifier.

2769 If the attribute identified by the attribute identifier is supported, and its associated selector field is valid, the device  
 2770 SHALL set the status field of the corresponding read attribute status record to SUCCESS and SHALL set the attribute  
 2771 value field to the value of the attribute (or its selected element). The device SHALL then move on to the next attribute  
 2772 identifier.

2773 **2.5.16 Write Attributes Structured Command**

2774 **2.5.16.1 Write Attributes Structured Command Frame Format**

2775 The Write Attributes Structured command frame SHALL be formatted as illustrated in Figure 2-31.

2776 **Figure 2-31. Write Attributes Structured Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Write attribute record 1	Write attribute record 2	...	Write attribute record <i>n</i>

2777 Each write attribute record SHALL be formatted as illustrated in Figure 2-32.

2778 **Figure 2-32. Format of the Write Attribute Record Field**

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

### 2779 **2.5.16.1.1 ZCL Header Fields**

2780 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2781 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write  
2782 Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific  
2783 attributes.

2784 The command identifier field SHALL be set to indicate the Write Attributes Structured command (see Table 2-3).

### 2785 **2.5.16.1.2 Attribute Identifier Field**

2786 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute that is to be written  
2787 (or an element of which is to be written).

### 2788 **2.5.16.1.3 Selector Field**

2789 The selector field specifies whether the whole of the attribute value is to be written, or only an individual element of  
2790 it. An individual element may only be written to attributes with types of Array, Structure, Set or Bag.

2791 The Selector field SHALL be formatted as illustrated in Figure 2-33.

2792 **Figure 2-33. Format of the Selector Field**

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

### 2793 **2.5.16.1.4 Writing an Element to an Array or Structure**

2794 When writing an element to an array or structure, the Indicator subfield indicates the number of index fields that follow  
2795 it. This number is limited to the range 0 - 15 (i.e., the upper 4 bits of the Indicator field are set to zero). It may be  
2796 further limited by an application.

2797 If the Indicator subfield is 0, there are no index fields, and the whole of the attribute value is to be written.

2798 If this subfield is 1 or greater, the index fields indicate which element is to be written, nested to a depth of  $m$ . For  
2799 example, if the attribute is an array of arrays (or structures), then if  $m = 2$ , index 1 = 5 and index 2 = 3, the third  
2800 element of the fifth element of the attribute will be written.

2801 Note that elements are numbered from 1 upwards for both arrays and structures.

2802 The zeroth element of an array or structure has type 16-bit unsigned integer, and holds the number of elements in the  
2803 array or structure. The zeroth element of an array may optionally be written (this is application dependent) and has  
2804 the effect of changing the number of elements of the array. If the number is reduced, the array is truncated. If the  
2805 number is increased, the content of new elements is application dependent.

2806 The zeroth element of a structure may not be written to. Writing to an element with an index greater than the number  
2807 of elements in an array or structure is always an error.

### 2808 **2.5.16.1.5 Adding/Removing an Element to/from a Set or Bag**

2809 This command may also be used to add an element to a set or bag, or to remove an element from a set or bag.

2810 In this case, the lower 4 bits of the Indicator subfield still indicate the number of index fields that follow it, as the set  
2811 may be an element of an array or structure, which may itself be nested inside other arrays or structures.

2812 The upper 4 bits of the Indicator subfield have the following values:

2813       0b0000     Write whole set/bag

2814       0b0001     Add element to the set/bag

2815       0b0010     Remove element from the set/bag

2816 All other values are reserved.

### 2817 **2.5.16.1.6 Attribute Data Type Field**

2818 The attribute data type field SHALL contain the data type of the attribute or element thereof that is to be written.

### 2819 **2.5.16.1.7 Attribute Value Field**

2820 The attribute value field is variable in length and SHALL contain the actual value of the attribute, or element thereof,  
2821 that is to be written. For an attribute or element of type array, structure, set or bag, this field has the same format as  
2822 for the Read Attributes Structured command (see Read Attributes Structured Command).

## 2823 **2.5.16.2 When Generated**

2824 The Write Attributes Structured command is generated when a device wishes to change the values of one or more  
2825 attributes located on another device. Each write attribute record SHALL contain the identifier and the actual value of  
2826 the attribute, or element thereof, to be written.

### 2827 **2.5.16.3 Effect on Receipt**

2828 On receipt of this command, the device SHALL attempt to process each specified write attribute record and SHALL  
2829 construct a write attribute structured response command. Each write attribute status record of the constructed  
2830 command SHALL contain the identifier from the corresponding write attribute record and a status value evaluated as  
2831 described below.

2832 For each write attribute record included in the command frame, the device SHALL first check that it corresponds to  
2833 an attribute that is implemented on this device and that its associated selector field correctly indicates either the whole  
2834 of the attribute or an element of the attribute. If it does not (e.g., an index is greater than the number of elements of an  
2835 array), the device SHALL set the status field of the corresponding write attribute status record to either  
2836 UNSUPPORTED\_ATTRIBUTE or INVALID\_SELECTOR as appropriate and move on to the next write attribute  
2837 record.

2838 If the attribute identified by the attribute identifier is supported, the device SHALL check whether the attribute data  
2839 type field is correct. (**Note:** If the element being written is the zeroth element of an array (to change the length of the  
2840 array) the data type must be 16-bit unsigned integer). If not, the device SHALL set the status field of the corresponding  
2841 write attribute status record to INVALID\_DATA\_TYPE and move on to the next write attribute record.

2842 If the attribute data type is correct, the device SHALL check whether the attribute is writable. If the attribute is  
2843 designated as read only, the device SHALL set the status field of the corresponding write attribute status record to  
2844 READ\_ONLY and move on to the next write attribute record. (**Note:** If an array may not have its length changed, its  
2845 zeroth element is read only).

2846 If the attribute is writable, the device SHALL check that all the supplied basic (e.g., integer, floating point) values in  
 2847 the attribute value field are within the specified ranges of the elements they are to be written to. If a supplied value  
 2848 does not fall within the specified range of its target element, the device SHALL set the status field of the corresponding  
 2849 write attribute status record to INVALID\_VALUE, SHALL set the selector field of that record to indicate that target  
 2850 element, and SHALL move on to the next write attribute record.

2851 The returned selector SHALL have the number of indices necessary to specify the specific low-level element that  
 2852 failed, which will be the same as or greater than the number of indices in the selector of the write attribute record.  
 2853 Note that if the element being written is the zeroth element of an array (to change the length of the array) and the  
 2854 requested new length is not acceptable to the application, the value being written is considered outside the specified  
 2855 range of the element.

2856 If the value supplied in the attribute value field is within the specified range of the attribute, the device SHALL proceed  
 2857 as follows.

- 2858 • If an element is being added to a set, and there is an element of the set that has the same value as the value to be  
 2859 added, the device SHALL set the status field of the corresponding write attribute status record to  
 2860 DUPLICATE\_ENTRY and move on to the next write attribute record.
- 2861 • Else, if an element is being removed from a set or a bag, and there is no element of the set or bag that has the  
 2862 same value as the value to be removed, the device SHALL set the status field of the corresponding write  
 2863 attribute status record to NOT\_FOUND and move on to the next write attribute record.
- 2864 • Otherwise, the device SHALL write, add or remove the supplied value to/from the identified attribute or  
 2865 element, as appropriate, and SHALL move on to the next write attribute record. In this (successful) case, a write  
 2866 attribute status record SHALL not be generated. (**Note:** If the element being written is the zeroth element of an  
 2867 array, the length of the array SHALL be changed. If the length is reduced, the array is truncated. If the length is  
 2868 increased, the content of new elements is application dependent.)

2869 When all write attribute records have been processed, the device SHALL generate the constructed Write Attributes  
 2870 Response command. If there are no write attribute status records in the constructed command, because all attributes  
 2871 were written successfully, a single write attribute status record SHALL be included in the command, with the status  
 2872 field set to SUCCESS and the attribute identifier field omitted.

## 2873 2.5.17 Write Attributes Structured Response Command

### 2874 2.5.17.1 Write Attributes Structured Response Command 2875 Frame Format

2876 The Write Attributes Response command frame SHALL be formatted as illustrated in Figure 2-34.

2877 **Figure 2-34. Write Attributes Structured Response Command Frame**

Octets: Variable	Variable	Variable	...	Variable
ZCL header	Write attribute status record 1	Write attribute status record 2	...	Write attribute status record <i>n</i>

2878

2879 Each write attribute status record SHALL be formatted as illustrated in Figure 2-35.

2880 **Figure 2-35. Format of the Write Attribute Status Record Field**

Octets: 1	2	Variable
Status	Attribute identifier	Selector



### 2881 **2.5.17.1.1 ZCL Header Fields**

2882 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
2883 command (0b00). The manufacturer specific sub-field SHALL be set to 0 if this command is being used to Write  
2884 Attributes defined for any cluster in the ZCL or 1 if this command is being used to write manufacturer specific  
2885 attributes.

2886 The command identifier field SHALL be set to indicate the Write Attributes Structured response command (see Table  
2887 2-3).

### 2888 **2.5.17.1.2 Status Field**

2889 The status field is 8 bits in length and specifies the status of the write operation attempted on this attribute, as detailed  
2890 in Effect on Receipt 2.5.16.3.

2891 Note that write attribute status records are not included for successfully written attributes, to save bandwidth. In the  
2892 case of successful writing of all attributes, only a single write attribute status record SHALL be included in the  
2893 command, with the status field set to SUCCESS and the attribute identifier and selector fields omitted.

### 2894 **2.5.17.1.3 Attribute Identifier Field**

2895 The attribute identifier field is 16 bits in length and SHALL contain the identifier of the attribute on which the write  
2896 operation was attempted.

### 2897 **2.5.17.1.4 Selector Field**

2898 The selector field SHALL specify the element of the attribute on which the write operation that failed was attempted.  
2899 See Figure 2-33 for the structure of this field.

2900 From the structure shown in Figure 2-33, note that for all attribute data types other than array or structure this field  
2901 consists of a single octet with value zero. For array or structure types, a single octet with value zero indicates that no  
2902 information is available about which element of the attribute caused the failure.

## 2903 **2.5.17.2 When Generated**

2904 The Write Attributes Structured response command is generated in response to a Write Attributes Structured  
2905 command.

## 2906 **2.5.17.3 Effect on Receipt**

2907 On receipt of this command, the device is notified of the results of its original Write Attributes Structured command.

## 2908 **2.5.18 Discover Commands Received Command**

2909 This command may be used to discover all commands processed (received) by this cluster, including optional or  
2910 manufacturer-specific commands.

### 2911 **2.5.18.1 Discover Commands Received Command Frame 2912 Format**

2913 The discover server commands command frame SHALL be formatted as follows.

2914 **Figure 2-36. Format of the Discover Server Commands Command Frame**

<b>Octets:</b>	Variable	1	1
<b>Field:</b>	ZCL header	Start command identifier	Maximum command identifiers

2915

2916 **2.5.18.1.1 ZCL Header Fields**

2917 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a  
 2918 global command (0b00). The manufacturer-specific sub-field SHALL be set to 0 to discover standard commands in a  
 2919 cluster or 1 to discover manufacturer-specific commands in either a standard or a manufacturer-specific cluster. A  
 2920 manufacturer ID in this field of 0xffff (wildcard) will discover any manufacture-specific commands. The direction  
 2921 bit SHALL be 0 (client to server) to discover commands that the server can process. The direction bit SHALL be 1  
 2922 (server to client) to discover commands that the client can process.

2923 The command identifier field SHALL be set to indicate the Discover Commands Received command.

2924 **2.5.18.1.2 Start Command Identifier Field**

2925 The start command identifier field is 8-bits in length and specifies the value of the identifier at which to begin the  
 2926 command discovery.

2927 **2.5.18.1.3 Maximum Command Identifiers Field**

2928 The maximum command identifiers field is 8-bits in length and specifies the maximum number of command identifiers  
 2929 that are to be returned in the resulting Discover Commands Received Response.

2930 **2.5.18.2 When Generated**

2931 The Discover Commands Received command is generated when a remote device wishes to discover the optional and  
 2932 mandatory commands the cluster to which this command is sent can process.

2933 **2.5.18.3 Effect on Receipt**

2934 On receipt of this command, the device SHALL construct an ordered list of command identifiers. This list SHALL  
 2935 start with the first command that has an identifier that is equal to or greater than the identifier specified in the start  
 2936 command identifier field. The number of command identifiers included in the list SHALL not exceed that specified  
 2937 in the maximum command identifiers field.

2938 **2.5.19 Discover Commands Received Response**

2939 The Discover Commands Received Response command is sent in response to a Discover Commands Received  
 2940 command, and is used to discover which commands a cluster can process.

2941 **2.5.19.1 Discover Commands Received Response Frame**

2942 The Discover Commands Received Response command frame SHALL be formatted as shown below:

2943 **Figure 2-37. Format of the Discover Commands Received Response Frame**

<b>Octets:</b>	Variable	1	1	1	...	1
<b>Field:</b>	ZCL Header	Discovery complete	Command identifier 1	Command identifier 2	...	Command identifier n

2944

2945 **2.5.19.1.1 ZCL Header Fields**

2946 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 2947 command (0b00). The manufacturer-specific sub-field SHALL be set to the same value included in the original  
 2948 discover commands command, with the exception that if the manufacture ID is 0xffff (wildcard), then the response  
 2949 will contain the manufacture ID of the manufacturer-specific commands, or will not be present if the cluster supports  
 2950 no manufacturer-specific extensions, or the manufacturer wishes to hide the fact that it supports extensions. The  
 2951 command identifier field SHALL be set to indicate the Discover Commands Received Response command.

2952 **2.5.19.1.2 Discovery Complete Field**

2953 The discovery complete field is a boolean field. A value of 0 indicates that there are more commands to be discovered.  
 2954 A value of 1 indicates that there are no more commands to be discovered.

2955 **2.5.19.1.3 Command Identifier Field**

2956 The command identifier field SHALL contain the identifier of a discovered command. Commands SHALL be  
 2957 included in ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute  
 2958 identifier field of the received discover server commands command.

2959 **2.5.19.2 When Generated**

2960 The Discover Commands Received Response is generated in response to a Discover Commands Received command.

2961 **2.5.19.3 Effect on Receipt**

2962 On receipt of this command, the device is notified of the results of its command discovery request. Following the  
 2963 receipt of this command, if the discovery complete field indicates that there are more commands to be discovered, the  
 2964 device may choose to send subsequent discover command request commands to obtain the rest of the command  
 2965 identifiers. In this case, the start command identifier specified in the next command discovery request command  
 2966 SHOULD be set equal to one plus the last command identifier received in the Discover Commands Received  
 2967 Response.

2968 **2.5.20 Discover Commands Generated Command**

2969 This command may be used to discover all commands which may be generated (sent) by the cluster, including optional  
 2970 or manufacturer-specific commands.

2971 **2.5.20.1 Discover Commands Generated Command Frame Format**

2972  
 2973 Except for the command ID in the ZCL header, the Discover Commands Generated command frame SHALL be  
 2974 formatted as described in sub-clause 2.5.18 and its subsections.

## 2975 **2.5.20.2 When Generated**

2976 The Discover Commands Generated command is generated when a remote device wishes to discover the commands  
2977 that a cluster may generate on the device to which this command is directed.

## 2978 **2.5.20.3 Effect on Receipt**

2979 On receipt of this command, the device SHALL construct an ordered list of command identifiers. This list SHALL  
2980 start with the first command that has an identifier that is equal to or greater than the identifier specified in the start  
2981 command identifier field. The number of command identifiers included in the list SHALL not exceed that specified  
2982 in the maximum command identifiers field.

## 2983 **2.5.21 Discover Commands Generated Response**

2984 The Discover Commands Generated Response command is sent in response to a Discover Commands Generated  
2985 command, and is used to discover which commands a cluster supports.

### 2986 **2.5.21.1 Discover Commands Generated Response Frame**

2987 Except for the command ID in the ZCL header, the Discover Commands Generated Response command frame SHALL  
2988 be formatted as described in sub-clause 2.5.18 and its subsections.

### 2989 **2.5.21.2 When Generated**

2990 The Discover Commands Generated Response is generated in response to a Discover Commands Generated  
2991 command.

### 2992 **2.5.21.3 Effect on Receipt**

2993 On receipt of this command, the device is notified of the results of its Discover Commands Generated command.  
2994 Following the receipt of this command, if the discovery complete field indicates that there are more commands to be  
2995 discovered, the device may choose to send subsequent Discover Commands Generated commands to obtain the rest  
2996 of the command identifiers. In this case, the start command identifier specified in the next Discover Commands  
2997 Generated command SHOULD be set equal to one plus the last command identifier received in the Discover  
2998 Commands Generated Response.

## 2999 **2.5.22 Discover Attributes Extended Command**

3000 This command is similar to the discover attributes command, but also includes a field to indicate whether the attribute  
3001 is readable, writeable or reportable.

### 3002 **2.5.22.1 Discover Attributes Extended Command Frame** 3003 **Format**

3004 The Discover Attributes Extended command frame SHALL be formatted as illustrated as follows.

3005 **Figure 2-38. Format of the Discover Attributes Extended Command Frame**

<b>Octets:</b>	Variable	2	1
<b>Field:</b>	ZCL Header	Start attribute identifier	Maximum attribute identifiers

3006

3007 **2.5.22.1.1 ZCL Header Fields**

3008 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
 3009 command (0b00). The manufacturer-specific sub-field SHALL be set to 0 to discover standard attributes in a cluster  
 3010 or 1 to discover manufacturer-specific attributes in either a standard or a manufacturer-specific cluster. A manufacturer  
 3011 ID in this field of 0xffff (wildcard) will discover any manufacture-specific attributes. The direction bit SHALL be 0  
 3012 (client to server) to discover attributes that the server hosts. The direction bit SHALL be 1 (server to client) to discover  
 3013 attributes that the client may host.

3014 The command identifier field SHALL be set to indicate the Discover Attributes Extended command.

3015 **2.5.22.1.2 Start Attribute Identifier Field**

3016 The start attribute identifier field is 16-bits in length and specifies the value of the identifier at which to begin the  
 3017 attribute discovery.

3018 **2.5.22.1.3 Maximum Attribute Identifiers Field**

3019 The maximum attribute identifiers field is 8 bits in length and specifies the maximum number of attribute identifiers  
 3020 that are to be returned in the resulting Discover Attributes Extended Response command.

3021 **2.5.22.2 When Generated**

3022 The Discover Attributes Extended command is generated when a remote device wishes to discover the identifiers and  
 3023 types of the attributes on a device which are supported within the cluster to which this command is directed, including  
 3024 whether the attribute is readable, writeable or reportable.

3025 **2.5.22.3 Effect on Receipt**

3026 On receipt of this command, the device SHALL construct an ordered list of attribute information records, each  
 3027 containing a discovered attribute identifier and its data type, in ascending order of attribute identifiers. This list SHALL  
 3028 start with the first attribute that has an identifier that is equal to or greater than the identifier specified in the start  
 3029 attribute identifier field. The number of attribute identifiers included in the list SHALL not exceed that specified in  
 3030 the maximum attribute identifiers field.

3031 **2.5.23 Discover Attributes Extended Response Command**

3032 This command is sent in response to a Discover Attributes Extended command, and is used to determine if attributes  
 3033 are readable, writable or reportable.

3034 **2.5.23.1 Discover Attributes Extended Response Command  
 3035 Frame Format**

3036 The Discover Attributes Extended Response command frame SHALL be formatted as illustrated as follows.

3037 **Figure 2-39. Format of the Discover Attributes Extended Response Command Frame**

<b>Octets:</b>	Variable	1	4	4	...	4
<b>Field:</b>	ZCL header	Discovery complete	Extended attribute information 1	Extended attribute information 2	...	Extended attribute information n

3038 Each extended attribute information field SHALL be formatted as follows.

3039 **Figure 2-40. Format of the Extended Attribute Information Fields**

<b>Octets:</b>	2	1	1
<b>Field:</b>	Attribute identifier	Attribute data type	Attribute access control

3040

### 3041 **2.5.23.1.1 ZCL Header Fields**

3042 The frame control field SHALL be specified as follows. The frame type sub-field SHALL be set to indicate a global  
3043 command (0b00). The manufacturer-specific sub-field SHALL be set to the same value included in the original  
3044 Discover Attributes Extended command, with the exception that if the manufacture ID is 0xffff (wildcard), then the  
3045 response will contain the manufacture ID of the manufacturer-specific attributes, or will not be present if the cluster  
3046 supports no manufacturer-specific extensions.

3047 The command identifier field SHALL be set to indicate the Discover Attributes Extended Response command.

### 3048 **2.5.23.1.2 Discovery Complete Field**

3049 The discovery complete field is a boolean field. A value of 0 indicates that there are more attributes to be discovered.  
3050 A value of 1 indicates that there are no more attributes to be discovered.

### 3051 **2.5.23.1.3 Attribute Identifier Field**

3052 The attribute identifier field SHALL contain the identifier of a discovered attribute. Attributes SHALL be included in  
3053 ascending order, starting with the lowest attribute identifier that is greater than or equal to the start attribute identifier  
3054 field of the received discover attributes command.

### 3055 **2.5.23.1.4 Attribute Data Type Field**

3056 The attribute data type field SHALL contain the data type of the attribute.

### 3057 **2.5.23.1.5 Attribute Access Control Field**

3058 The attribute access control field SHALL indicate whether the attribute is readable, writable, and/or reportable. This  
3059 is an 8-bit bitmask field as shown below: The bits are in little endian order (bit 0 is listed first).

3060 **Figure 2-41. Format of the Attribute Access Control Field**

<b>Bits:</b>	1	1	1
<b>Field:</b>	Readable	Writeable	Reportable

3061

### 3062 **2.5.23.2 When Generated**

3063 The Discover Attributes Extended Response command is generated in response to a Discover Attributes Extended  
3064 command.

3065 **2.5.23.3 Effect on Receipt**

3066 On receipt of this command, the device is notified of the results of its Discover Attributes Extended command.  
 3067 Following the receipt of this command, if the discovery complete field indicates that there are more attributes to be  
 3068 discovered, the device may choose to send subsequent Discover Attributes Extended commands to obtain the rest of  
 3069 the attribute identifiers and access control. In this case, the start attribute identifier specified in the next Discover  
 3070 Attributes Extended command SHOULD be set equal to one plus the last attribute identifier received in the Discover  
 3071 Attributes Extended Response command.

3072 **2.6 Addressing, Types and Enumerations**

3073 **2.6.1 Addressing**

3074 The architecture uses a number of concepts to address applications, clusters, device descriptions, attributes and  
 3075 commands, each with their own constraints. This sub-clause details these constraints.

3076 **2.6.1.1 Profile Identifier**

3077 A profile identifier is 16 bits in length and specifies the application profile being used. A profile identifier SHALL be  
 3078 set to one of the non-reserved values listed in Table 2-5. Please see [Z5], Application Architecture for more details.

3079 **Table 2-5. Valid Profile Identifier Values**

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

3080 **2.6.1.2 Device Identifier**

3081 A device identifier is 16 bits in length and specifies a specific device within a standard. A device identifier SHALL  
 3082 be set to one of the non-reserved values listed in Table 2-6. Please see [Z5], Application Architecture for more details.

3083 **Table 2-6. Valid Device Identifier Values**

Device Identifier	Description
0x0000 – 0xbfff	Standard device description.
<i>all other values</i>	Reserved

3084 **2.6.1.3 Cluster Identifier**

3085 A cluster identifier is 16 bits in length and identifies an instance of an implemented cluster specification (see 2.2.1.1).  
 3086 It SHALL be set to one of the non-reserved values listed in Table 2-7. Please see [Z5], Application Architecture for  
 3087 more details.

3088

Table 2-7. Valid Cluster Identifier Values

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

3089 **2.6.1.4 Attribute Identifier**

3090 An attribute identifier is 16 bits in length and specifies a single attribute within a cluster. An attribute identifier, defined  
 3091 within the ZCL, SHALL be set to one of the non-reserved values listed in Table 2-8. Undefined cluster attributes are  
 3092 reserved for future cluster attributes. Global attributes are associated with all clusters (see 2.3.4.3).

3093

Table 2-8. Valid ZCL Defined Attribute Identifier Values

Attribute Identifier	Description
0x0000 – 0x4fff	Standard attribute
0xf000 – 0xffff	Global Attributes
<i>all other values</i>	Reserved

3094

3095 Manufacturer specific attributes within a standard cluster can be defined over the full 16-bit range. These may be  
 3096 manipulated using the global commands listed in Table 2-3, with the frame control field set to indicate a manufacturer  
 3097 specific command (see 2.4). (Note that, alternatively, the manufacturer may define his own cluster specific commands  
 3098 (see 2.4), re-using these command IDs if desired).

3099 **2.6.1.5 Command Identifier**

3100 A command identifier is 8 bits in length and specifies a global command within the ZCL or a cluster specific command.  
 3101 A command identifier SHALL be set to one of the non-reserved values listed in Table 2-9. Manufacturer specific  
 3102 commands within a standard cluster can be defined over the full 8-bit range but each SHALL use the appropriate  
 3103 manufacturer code.

3104

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

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



3105 **2.6.2 Data Types**

3106 Each attribute and command field in a cluster specification SHALL have a well-defined data type. Each attribute in a  
 3107 cluster specification SHALL map to a single data type identifier (data type ID), which describes the length and general  
 3108 properties of the data type.

3109 New data type identifiers SHALL NOT be added to this table. It is encouraged that commonly used cluster attribute  
 3110 and command field data types are added to this list and mapped to appropriate data type identifiers with a unique  
 3111 name. Such common data types can then be reused instead of redefined in each specification. For example: a  
 3112 percentage data type representing 0-100% with a unit size of .5 percent, would be mapped to data type identifier 0x20  
 3113 (also unsigned 8-bit integer), and perhaps named ‘Percent 8-bit .5-unit’ (short named ‘percent8.5’).

3114 The table also indicates for each data type whether it defines an analog or discrete value. Values of analog types may  
 3115 be added to or subtracted from other values of the same type, and are typically used to measure the value of properties  
 3116 in the real world that vary continuously over a range. Values of discrete data types only have meaning as individual  
 3117 values, and may not be added or subtracted.

3118 Cluster specifications SHALL use the unique data type short name to reduce the text size of the specification.

3119 The Invalid Value is only specified when an invalid data value is required for an attribute. The Invalid Value for an  
 3120 analog data type MAY be within the valid data range as defined in an attribute specification.<sup>6</sup>

3121 **Table 2-10. Data Types**

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

<sup>6</sup> CCB 2213 explain about invalid values

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

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

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

### 3122 **2.6.2.1 No Data Type**

3123 The no data type is a special type to represent an attribute with no associated data.

### 3124 **2.6.2.2 General Data (8, 16, 24, 32, 40, 48, 56 and 64-bit)**

3125 This type has no rules about its use, and may be used when a data element is needed but its use does not conform to  
3126 any of the standard types.

### 3127 **2.6.2.3 Boolean**

3128 The Boolean type represents a logical value, either FALSE (0x00) or TRUE (0x01). The value 0xff represents an  
3129 invalid value of this type. All other values of this type are forbidden.

### 3130 **2.6.2.4 Bitmap (8, 16, 24, 32, 40, 48, 56 and 64-bit)**

3131 The Bitmap type holds 8, 16, 24, 32, 40, 48, 56 or 64 logical values, one per bit, depending on its length. There is no  
3132 value that represents an invalid value of this type.

### 3133 **2.6.2.5 Unsigned Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit)**

3134 This type represents an unsigned integer with a decimal range of 0 to  $2^8-1$ , 0 to  $2^{16}-1$ , 0 to  $2^{24}-1$ , 0 to  $2^{32}-1$ , 0 to  $2^{40}-1$ ,  
3135 0 to  $2^{48}-1$ , 0 to  $2^{56}-1$ , or 0 to  $2^{64}-1$ , depending on its length. The values that represent an invalid value of this type are  
3136 0xff, 0xffff, 0xfffffff, 0xffffffffff, 0xffffffffff, 0xffffffffff, 0xffffffffff, 0xffffffffff and 0xffffffffff respectively.

### 3137 **2.6.2.6 Signed Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit)**

3138 This type represents a signed integer with a decimal range of  $-(2^7-1)$  to  $2^7-1$ ,  $-(2^{15}-1)$  to  $2^{15}-1$ ,  $-(2^{23}-1)$  to  $2^{23}-1$ ,  $-(2^{31}-1)$   
3139 to  $2^{31}-1$ ,  $-(2^{39}-1)$  to  $2^{39}-1$ ,  $-(2^{47}-1)$  to  $2^{47}-1$ ,  $-(2^{55}-1)$  to  $2^{55}-1$ , or  $-(2^{63}-1)$  to  $2^{63}-1$ , depending on its length. The values that  
3140 represent an invalid value of this type are 0x80, 0x8000, 0x800000, 0x80000000, 0x8000000000, 0x800000000000,  
3141 0x80000000000000 and 0x8000000000000000 respectively.

### 3142 **2.6.2.7 Enumeration (8-bit, 16-bit)**

3143 The Enumeration type represents an index into a lookup table to determine the final value. The values 0xff and 0xffff  
3144 represent invalid values of the 8-bit and 16-bit types respectively.

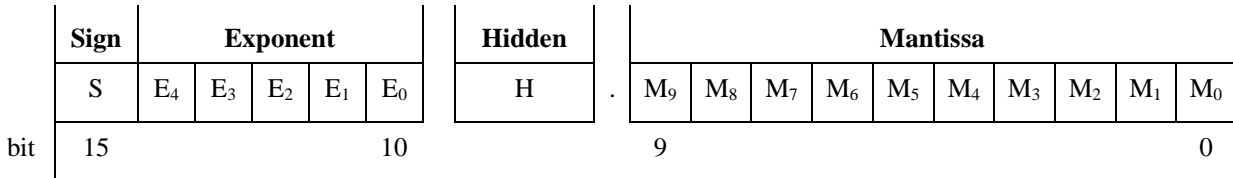
### 3145 **2.6.2.8 Semi-precision**

3146 The semi-precision number format is based on the IEEE 754 standard for binary floating-point arithmetic [E2]. This  
3147 number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing required  
3148 supporting it.

3149 The value is calculated as:

3150 
$$\text{Value} = -1^{\text{Sign}} * (\text{Hidden} + \text{Mantissa}/1024) * 2^{(\text{Exponent}-15)}$$

3151 **Figure 2-42. Format of the Semi-precision Number**



3152 **Note:** The transmission order for the format in Figure 2-42 is bit 0 first.

3153 For normalized numbers ( $>2^{-14}$ ), the hidden bit = 1 and the resolution is constant at 11 bits (1 in 2048).

3154 For un-normalized numbers, the hidden bit = 0. Note that this does not maintain 11-bit resolution and that the  
 3155 resolution becomes coarser as the number gets smaller.

3156 The hidden bit is not sent over the link. It SHALL have the value ‘1’ (i.e., normalized) in order to be classified as a  
 3157 semi-precision number.

3158 The sign bit is set to 0 for positive values, 1 for negative.

3159 The exponent is 5 bits. The actual exponent of 2 is calculated as (exponent – 15).

3160 Certain values are reserved for specific purposes:

- 3161 • **Not a Number:** this is used for undefined values (e.g., at switch-on and before initialization) and is indicated by  
 3162 an exponent of 31 with a non-zero mantissa.
- 3163 • **Infinity:** this is indicated by an exponent of 31 and a zero mantissa. The sign bit indicates whether this  
 3164 represents + infinity or – infinity, the figure of 0x7c00 representing  $+\infty$  and 0xfc00 representing  $-\infty$ .
- 3165 • **Zero:** this is indicated by both a zero exponent and zero mantissa. The sign bit indicates whether this is + or –  
 3166 zero, the value 0x0000 representing +zero and 0x8000 representing –zero.
- 3167 • **Un-normalized numbers:** numbers  $< 2^{-14}$  are indicated by a value of 0 for the exponent. The hidden bit is set to  
 3168 zero.

3169 The maximum value represented by the mantissa is 0x3ff / 1024. The largest number that can be represented is  
 3170 therefore:

3171 
$$-1^{\text{Sign}} * (1 + 1023/1024) * 2^{(30-15)} = \pm 1.9990234 * 32768 = \pm 65504$$

3172 Certain applications may choose to scale this value to allow representation of larger values (with a correspondingly  
 3173 coarser resolution). For details, see the relevant device descriptions.

3174 For example, a value of +2 is represented by  $+2^{(16-15)} * 1.0 = 0x4000$ , while a value of –2 is represented by 0xc000.

3175 Similarly, a value of +0.625 is represented by  $+2^{(17-15)} * 1.625 = 0x4680$ , while –0.625 is represented by 0xc680.

### 3176 2.6.2.9 Single Precision

3177 The format of the single precision data type is based on the IEEE 754 standard for binary floating-point arithmetic  
 3178 [E2]. This number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing  
 3179 required supporting it.

3180 The format and interpretation of values of this data type follow the same rules as given for the semi-precision data  
 3181 type, but with longer sub-fields, as follows.

3182 Length of mantissa = 23 bits, length of exponent = 8 bits

3183 For further details, see [E2].

### 3184 2.6.2.10 Double Precision

3185 The format of the double precision data type is based on the IEEE 754 standard for binary floating-point arithmetic  
3186 [E2]. This number format SHOULD be used very sparingly, when necessary, keeping in mind the code and processing  
3187 required supporting it.

3188 The format and interpretation of values of this data type follow the same rules as given for the semi-precision data  
3189 type, but with longer sub-fields, as follows.

3190 Length of mantissa = 52 bits, length of exponent = 11 bits

3191 For further details, see [E2].

### 3192 2.6.2.11 Octet String

3193 The octet string data type contains data in an application-defined format, not defined in this specification. The octet  
3194 string data type is formatted as illustrated in Figure 2-43.

3195 **Figure 2-43. Format of the Octet String Type**

Octets: 1	Variable
Octet count	Octet data

3196 The octet count sub-field is one octet in length and specifies the number of octets contained in the octet data sub-field.

3197 Setting this sub-field to 0x00 represents an octet string with no octet data (an “empty string”). Setting this sub-field to  
3198 0xff represents an invalid octet string value. In both cases the octet data sub-field has zero length.

3199 The octet data sub-field is  $n$  octets in length, where  $n$  is the value of the octet count sub-field. This sub-field contains  
3200 the application-defined data.

### 3201 2.6.2.12 Character String

3202 The character string data type contains data octets encoding characters according to the language and character set  
3203 field of the complex descriptor (see [Z1]). If not specified by the complex descriptor, the default character encoding  
3204 SHALL be UTF-8. The character string data type SHALL be formatted as illustrated in Figure 2-44.

3205 **Figure 2-44. Format of the Character String Type**

Octets: 1	Variable
Character data length	Character data

3206 The character data length sub-field is one octet in length and specifies the length of the character data sub-field. (**Note:**  
3207 for the ISO 646 ASCII character set, this is the same as the number of characters in the string. For other codings, this  
3208 may not be the case.)

3209 Setting this sub-field to 0x00 represents a character string with no character data (an “empty string”). Setting this sub-  
3210 field to 0xff represents an invalid character string value. In both cases the character data sub-field has zero length.

3211 The character data sub-field contains the encoded characters that comprise the desired character string. Its length is  
3212 the sum of the lengths of the characters as specified by the language and character set fields of the complex descriptor.

3213 A character string with no contents, i.e., with the character count sub-field equal to 0x00 and a zero length character  
3214 data sub-field, SHALL be referred to as an 'empty string'.

### 3215 **2.6.2.13 Long Octet String**

3216 The long octet string data type contains data in an application-defined format, not defined in this specification. The  
 3217 long octet string data type is formatted as illustrated in Figure 2-45.

3218 **Figure 2-45. Format of the Long Octet String Type**

Octets: 2	Variable
Octet count	Octet data

3219  
 3220 The octet count sub-field is two octets in length and specifies the number of octets contained in the octet data sub-  
 3221 field. It has the same format as a 16-bit unsigned integer (see 2.6.2.5).

3222 Setting this sub-field to 0x0000 represents a long octet string with no octet data (an “empty string”). Setting this sub-  
 3223 field to 0xffff represents an invalid long octet string value. In both cases the octet data sub-field has zero length.

3224 The octet data sub-field is *n* octets in length, where *n* is the value of the octet count sub-field. This sub-field contains  
 3225 the application-defined data.

### 3226 **2.6.2.14 Long Character String**

3227 The long character string data type contains data octets encoding characters according to the language and character  
 3228 set field of the complex descriptor (see [Z1]). If not specified by the complex descriptor, the default character encoding  
 3229 SHALL be UTF-8. The long character string data type is formatted as illustrated in Figure 2-46.

3230 **Figure 2-46. Format of the Long Character String Type**

Octets: 2	Variable
Character count	Character data

3231  
 3232 The character count sub-field is two octets in length and specifies the length of the character data sub-field. (**Note:** for  
 3233 the ISO 646 ASCII character set, this is the same as the number of characters in the string. For other codings, this may  
 3234 not be the case.) It has the same format as a 16-bit unsigned integer (see 2.6.2.5).

3235 Setting this sub-field to 0x0000 represents a long character string with no character data (an “empty string”). Setting  
 3236 this sub-field to 0xffff represents an invalid long character string value. In both cases the character data sub-field has  
 3237 zero length.

3238 The character data sub-field contains the encoded characters that comprise the desired character string. Its length is  
 3239 the sum of the lengths of the characters as specified by the language and character set fields of the complex descriptor.

3240 A character string with no contents, i.e., with the character count sub-field equal to 0x0000 and a zero length character  
 3241 data sub-field, SHALL be referred to as an 'empty string'.

### 3242 **2.6.2.15 Fixed ASCII**

3243 This data type is defined for legacy reasons, so that there is a data type to represent an ASCII display string that has a  
 3244 fixed length as defined in the specification. The NUL ASCII character 0x00 shall terminate the string, unless the string  
 3245 takes up the entire fixed length.

3246 This data type SHALL NOT be used as a data type for an attribute, because it does not have an associated length, nor  
 3247 Data Type Id. It is not recommended to use this data type, when a more well-defined data type exists.

### 3248 **2.6.2.16 Array**

3249 An array is an ordered sequence of zero or more elements, all of the same data type. This data type may be any ZCL  
3250 defined data type, including array, structure, bag or set. The total nesting depth is limited to 15, and may be further  
3251 limited by an application.

3252 Individual elements may be accessed by an index of type 16-bit unsigned integer. Elements are numbered from 1  
3253 upwards. The zeroth element is readable, always has type uint16, and holds the number of elements contained in the  
3254 array, which may be zero. If the zeroth element contains 0xffff, the array is considered invalid / undefined.

3255 The zeroth element may also, as an implementation option, be writable, to change the size of the array (see 2.5.16.1  
3256 for details).

3257 Arrays are 'packed', i.e., there is no concept of a 'null' element. However, if an element has a simple (unstructured)  
3258 type, and that type has an 'invalid number' value defined, then that value indicates that the element is invalid /  
3259 undefined.

### 3260 **2.6.2.17 Structure**

3261 A structure is an ordered sequence of elements, which may be of different data types. Each data type may be any ZCL  
3262 defined data type, including array, structure, bag or set. The total nesting depth is limited to 15, and may be further  
3263 limited by an application.

3264 Individual elements may be accessed by an index of type 16-bit unsigned integer. Elements are numbered from 1  
3265 upwards. The zeroth element is readable, always has type 16-bit unsigned integer, and holds the number of elements  
3266 contained in the structure, which may be zero. If the zeroth element contains 0xffff, the array is considered invalid /  
3267 undefined. The zeroth element may not be written to.

3268 Structures are 'packed', i.e., there is no concept of a 'null' element. However, if an element has a simple (unstructured)  
3269 type, and that type has an 'invalid number' value defined, that value indicates that the element is undefined.

### 3270 **2.6.2.18 Set**

3271 A set is a collection of elements with no associated order. Each element has the same data type, which may be any  
3272 ZCL defined data type, including array, structure, bag or set. The nesting depth is limited to 15, and may be further  
3273 limited by an application.

3274 Elements of a set are not individually addressable, so may not be individually read or modified. Sets may only be read  
3275 in their entirety. Individual elements may be added to a set or removed from a set; removal is done by value.

3276 The maximum number of elements in a set is 0xfffe. If the number of elements is returned by a read command as  
3277 0xffff, this indicates that it is invalid / undefined.

3278 No two elements of a set may have the same value.

### 3279 **2.6.2.19 Bag**

3280 A bag behaves the same as a set, except that the restriction that no two elements may have the same value is removed.

### 3281 **2.6.2.20 Time of Day**

3282 The Time of Day data type SHALL be formatted as illustrated in Figure 2-47.



3283

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

<b>Octets: 1</b>	<b>1</b>	<b>1</b>	<b>1</b>
Hours	Minutes	Seconds	Hundredths

3284

3285 The hours subfield represents hours according to a 24-hour clock. The range is from 0 to 23.

3286 The minutes subfield represents minutes of the current hour. The range is from 0 to 59.

3287 The seconds subfield represents seconds of the current minute. The range is from 0 to 59.

3288 The hundredths subfield represents 100ths of the current second. The range is from 0 to 99.

3289 A value of 0xff in any subfield indicates an unused subfield. If all subfields have the value 0xff, this indicates an  
 3290 invalid or 'don't care' value of the data type.

### 3291 **2.6.2.21 Date**

3292 The Date data type SHALL be formatted as illustrated in Figure 2-48.

3293

**Figure 2-48. Format of the Date Type**

<b>Octets: 1</b>	<b>1</b>	<b>1</b>	<b>1</b>
Year - 1900	Month	Day of month	Day of week

3294

3295 The year - 1900 subfield has a range of 0 to 255, representing years from 1900 to 2155.

3296 The month subfield has a range of 1 to 12, representing January to December.

3297 The day of month subfield has a range of 1 to 31. Note that values in the range 29 to 31 may be invalid, depending on  
 3298 the month and year.

3299 The day of week subfield has a range of 1 to 7, representing Monday to Sunday.

3300 A value of 0xff in any subfield indicates an unused subfield. If all subfields have the value 0xff, this indicates an  
 3301 invalid or 'don't care' value of the data type.

### 3302 **2.6.2.22 UTCTime**

3303 UTCTime is an unsigned 32-bit value representing the number of seconds since 0 hours, 0 minutes, 0 seconds, on the  
 3304 1st of January, 2000 UTC (Universal Coordinated Time). The value that represents an invalid value of this type is  
 3305 0xffffffff.

3306 Note that UTCTime does not hold a standard textual representation of Universal Coordinated Time (UTC). However,  
 3307 UTC (to a precision of one second) may be derived from it.

### 3308 **2.6.2.23 Cluster ID**

3309 This type represents a cluster identifier as defined in 2.6.1.3.

### 3310 **2.6.2.24 Attribute ID**

3311 This type represents an attribute identifier as defined in 2.6.1.4.

### 3312 **2.6.2.25 BACnet OID (Object Identifier)**

3313 The BACnet OID data type is included to allow interworking with BACnet (see [A1]). The format is described in the  
3314 referenced standard.

### 3315 **2.6.2.26 IEEE Address**

3316 The IEEE Address data type is a 64-bit IEEE address that is unique to every node. A value of 0xffffffffffffff indicates  
3317 that the address is unknown.

### 3318 **2.6.2.27 128-bit Security Key**

3319 The 128-bit Security Key data type may take any 128-bit value.

### 3320 **2.6.2.28 Opaque**

3321 Fixed block or series of octets where the length is determined separately. The length SHALL be fixed in the  
3322 specification or determined from information from another part of the protocol. The format of the data MAY also be  
3323 unknown. It is not recommended to use this data type, when a more well-defined data type exists.

3324 This data type SHALL NOT be used as a cluster attribute, or have a Data Type Id.

### 3325 **2.6.2.29 Unknown**

3326 This data type SHALL NOT be used for a cluster attribute or frame data field. This is not an actual data type. It is  
3327 listed here for completeness and to reserve the data type identifier for use where one is required to designate that a  
3328 data type is unknown. It SHALL never be used to identify actual data as unknown. If the structure, format, or length  
3329 of data is unknown, or an existing data type cannot be used, then the Opaque data type SHALL be used.

## 3330 **2.6.3 Status Enumerations**

3331 Where a ZCL command contains a status field, the actual value of the enumerated status values is listed in Table 2-11.

3332 **Table 2-11. Enumerated Status Values Used in the ZCL**

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

<sup>7</sup> CCB 2318 a reserved field may be used in the future and then be non-zero

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

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

# CHAPTER 3 GENERAL

3334

3335 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 3336 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 3337 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 3338 using [*Rn*] notation.

## 3339 3.1 General Description

### 3340 3.1.1 Introduction

3341 The clusters specified in this document are generic measurement and sensing interfaces that are sufficiently general  
 3342 to be of use across a wide range of application domains.

### 3343 3.1.2 Cluster List

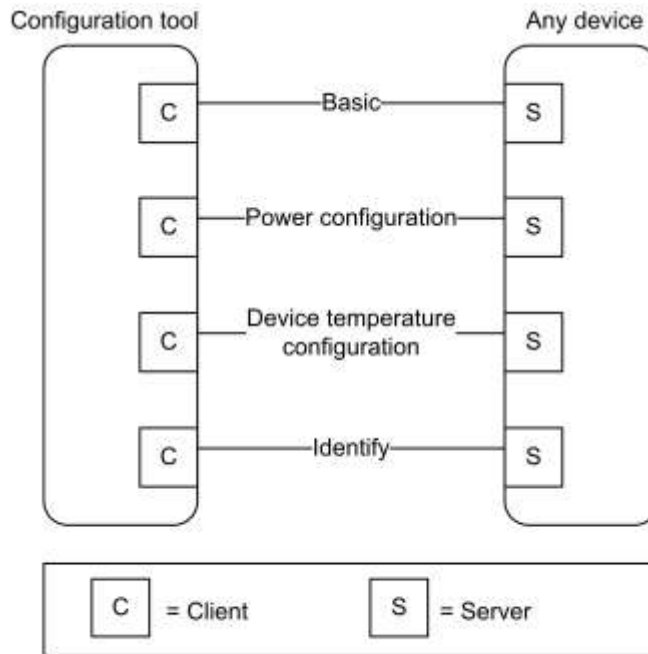
3344 This section lists the clusters specified in this document, and gives examples of typical usage.

3345 **Table 3-1. Device Configuration and Installation Clusters**

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

3346

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



*Note: Device names are examples for illustration purposes only*

3347

3348

**Table 3-2. Groups and Scenes Clusters**

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

3349

3350

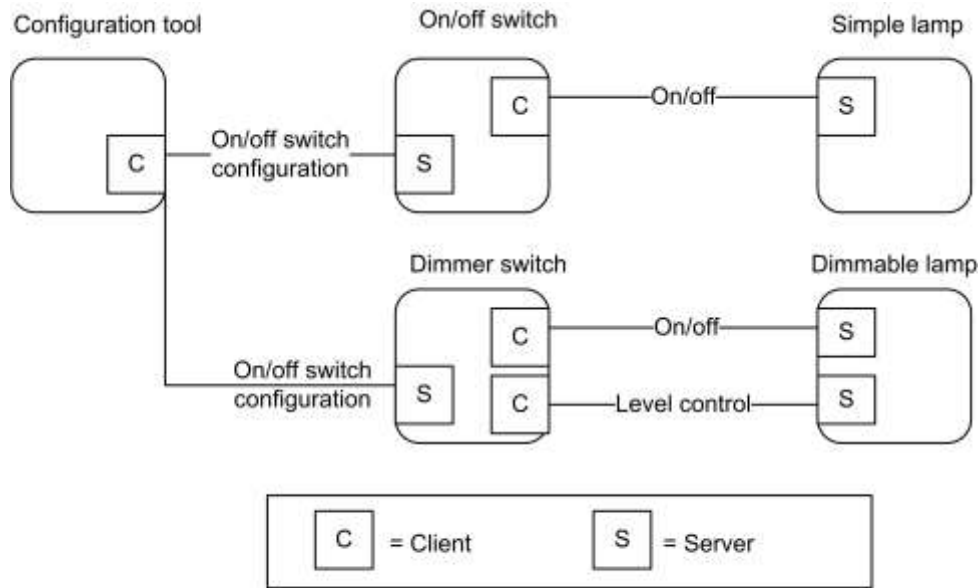
**Table 3-3. On/Off and Level Control Clusters**

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

3351

3352

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



*Note: Device names are examples for illustration purposes only*

3353

3354

3355

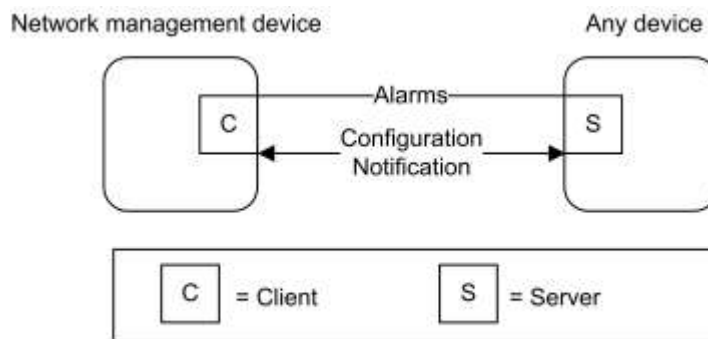
**Table 3-4. Alarms Cluster**

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

3356

3357

**Figure 3-3. Typical Usage of the Alarms Cluster**



*Note: Device names are examples for illustration purposes only*

3358

3359

3360

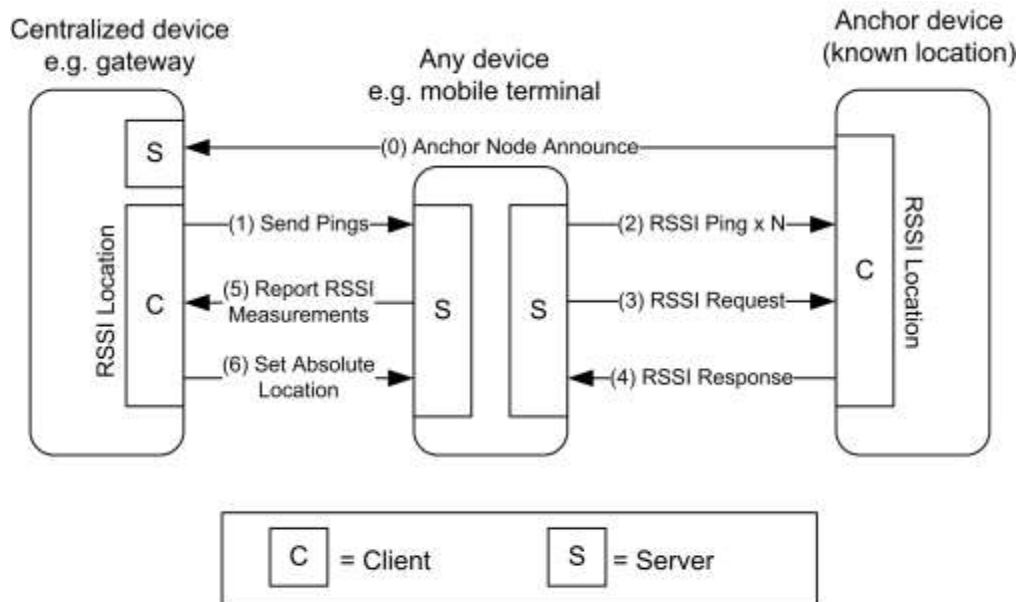
**Table 3-5. Other Clusters**

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

3361

3362

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



*Note: Device names are examples for illustration purposes only*

3363

3364

**Table 3-6. Generic Clusters**

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

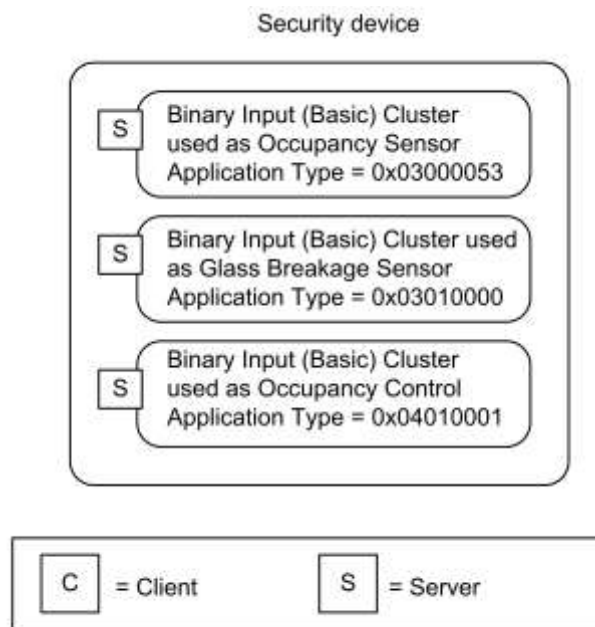


ID	Cluster Name	Description
0x000d	Analog Output (basic)	An interface for setting the value of an analog output (typically to the environment) and accessing various characteristics of that value.
0x000e	Analog Value (basic)	An interface for setting an analog value, typically used as a control system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

3365

3366

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



3367

*Note: Device names are examples for illustration purposes only*

3368 

## 3.2 Basic

3369 

### 3.2.1 Overview

3370 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
3371 etc.

3372 This cluster supports an interface to the node or physical device. It provides attributes and commands for determining  
3373 basic information, setting user information such as location, and resetting to factory defaults.

3374 **Note:** Where a node supports multiple endpoints, it will often be the case that many of these settings will apply to the  
3375 whole node, that is, they are the same for every endpoint on the node. In such cases, they can be implemented once  
3376 for the node, and mapped to each endpoint.

3377 

#### 3.2.1.1 Revision History

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

3378 

#### 3.2.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	B

3379 

#### 3.2.1.3 Cluster Identifiers

Identifier	Name
0x0000	Basic

3380 

## 3.2.2 Server

3381 

### 3.2.2.1 Dependencies

3382 For the alarms functionality of this cluster to be operational, the Alarms cluster server SHALL be implemented on the  
3383 same endpoint.

3384 

### 3.2.2.2 Attributes

3385 The Basic cluster attributes are summarized in Table 3-7.

3386 **Table 3-7. Attributes of the Basic Cluster**

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

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

3387 **3.2.2.2.1 ZCLVersion Attribute**

3388 The *ZCLVersion* attribute represents a published set of foundation items (in Chapter 2), such as global commands and  
 3389 functional descriptions. **For this version of the ZCL, this attribute SHALL be set to 0x03.**

3390 **3.2.2.2.2 ApplicationVersion Attribute**

3391 The *ApplicationVersion* attribute is 8 bits in length and specifies the version number of the application software  
 3392 contained in the device. The usage of this attribute is manufacturer dependent.

3393 **3.2.2.2.3 StackVersion Attribute**

3394 The *StackVersion* attribute is 8 bits in length and specifies the version number of the implementation of the stack  
 3395 contained in the device. The usage of this attribute is manufacturer dependent.

<sup>8</sup> ZLO 1.0

3396 **3.2.2.2.4 HWVersion Attribute**

3397 The *HWVersion* attribute is 8 bits in length and specifies the version number of the hardware of the device. The usage  
3398 of this attribute is manufacturer dependent.

3399 **3.2.2.2.5 ManufacturerName Attribute**

3400 The *ManufacturerName* attribute is a maximum of 32 bytes in length and specifies the name of the manufacturer as a  
3401 character string.

3402 **3.2.2.2.6 ModelIdentifier Attribute**

3403 The *ModelIdentifier* attribute is a maximum of 32 bytes in length and specifies the model number (or other identifier)  
3404 assigned by the manufacturer as a character string.

3405 **3.2.2.2.7 DateCode Attribute**

3406 The *DateCode* attribute is a character string with a maximum length of 16 bytes. The first 8 characters specify the date  
3407 of manufacturer of the device in international date notation according to ISO 8601, i.e., YYYYMMDD, e.g.,  
3408 20060814.

3409 The final 8 characters MAY include country, factory, line, shift or other related information at the option of the  
3410 manufacturer. The format of this information is manufacturer dependent.

3411 **3.2.2.2.8 PowerSource Attribute**

3412 The *PowerSource* attribute is 8 bits in length and specifies the source(s) of power available to the device. Bits  $b_0$ - $b_6$   
3413 of this attribute represent the primary power source of the device and bit  $b_7$  indicates whether the device has a  
3414 secondary power source in the form of a battery backup.

3415 This attribute SHALL be set to one of the non-reserved values listed in Table 3-8. Bit 7 of this attribute SHALL be  
3416 set to 1 if the device has a secondary power source in the form of a battery backup. Otherwise, bit 7 SHALL be set to  
3417 0.

3418 **Table 3-8. Values of the *PowerSource* Attribute**

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

<sup>9</sup> CCB 2229 add enum values to represent the ‘mapped’ bit that means battery backup

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

3419 **3.2.2.2.9 GenericDeviceClass Attribute**

3420 The *GenericDeviceClass* attribute defines the field of application of the *GenericDeviceType* attribute. It SHALL be  
 3421 set to one of the non-reserved values listed below:

3422

3423 **Table 3-9. Values of the *GenericDeviceClass* attribute**

<i>GenericDeviceClass</i> value	Description
0x00	Lighting
0x01 to 0xff	Reserved

3424 **3.2.2.2.10 GenericDeviceType Attribute**

3425 The *GenericDeviceType* attribute allows an application to show an icon on a rich user interface (e.g. smartphone app).

3426 Notes on the usage of the *GenericDeviceType* attribute:

- 3427 • lamps with integrated radio module SHALL have a proper value indicating the lamp type, according to the  
 3428 table below;
- 3429 • devices that cannot be assigned to a proper category SHALL be set as “unspecified”;

3430

3431 When the *GenericDeviceClass* attribute is set to 0x00 (i.e. lighting) the *GenericDeviceType* attribute SHALL be set  
 3432 to one of the non-reserved values listed below:

3433 **Table 3-10. Values of the *GenericDeviceType* attribute for the lighting class**

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

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

3434 **3.2.2.2.11 ProductCode Attribute**

3435 The *ProductCode* attribute allows an application to specify a code for the product. The *ProductCode* attribute SHALL  
3436 have the format defined in Figure .

3437

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

3438 **Figure 3-6. Format of the *ProductCode* attribute**

3439

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

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

3440 In case no code has been provided, the Octet Count field SHALL be set to 0 (i.e. the octet string is empty).

3441 **3.2.2.2.12 ProductURL Attribute**

3442 The *ProductURL* attribute specifies a link to a web page containing specific product information.

3443 Notes on the usage of the *ProductURL* attribute:

- 3444 • The length of the URL SHALL be limited by the maximum number of bytes that can be transmitted from
- 3445 the application in a single frame. In most cases, such limit is around 50 bytes.
- 3446 • In case no URL has been provided, the string SHALL be empty (i.e. the first byte is set to zero).

3447 **3.2.2.2.13 Manufacturer VersionDetails Attribute**

3448 Vendor specific human readable (displayable) string representing the versions of one of more program images  
 3449 supported on the device.

3450 **3.2.2.2.14 SerialNumber Attribute**

3451 Vendor specific human readable (displayable) serial number.

3452 **3.2.2.2.15 ProductLabel Attribute**

3453 Vendor specific human readable (displayable) product label.

3454 **3.2.2.2.16 LocationDescription Attribute**

3455 The *LocationDescription* attribute is a maximum of 16 bytes in length and describes the physical location of the device  
 3456 as a character string. This location description MAY be added into the device during commissioning.

3457 **3.2.2.2.17 PhysicalEnvironment Attribute**

3458 The *PhysicalEnvironment* attribute is 8 bits in length and specifies the type of physical environment in which the  
 3459 device will operate. This attribute SHALL be set to one of the non-reserved values listed in Table 3-12. All values are  
 3460 valid for endpoints supporting all profiles except when noted.

3461

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

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



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

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

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

3462 **3.2.2.2.18 DeviceEnabled Attribute**

3463 The *DeviceEnabled* attribute is a Boolean and specifies whether the device is enabled or disabled. This attribute  
 3464 SHALL be set to one of the non-reserved values listed in Table 3-13.

3465 **Table 3-13. Values of the *DeviceEnable* Attribute**

<i>DeviceEnable</i> Attribute Value	Description
0x00	Disabled
0x01	Enabled

3466 'Disabled' means that the device does not send or respond to application level commands, other than commands to  
 3467 read or write attributes. Values of attributes which depend on the operation of the application MAY be invalid, and  
 3468 any functionality triggered by writing to such attributes MAY be disabled. Networking functionality remains  
 3469 operational.

3470 If implemented, the identify cluster cannot be disabled, i.e., it remains functional regardless of this setting.

3471 **3.2.2.2.19 AlarmMask Attribute**

3472 The *AlarmMask* attribute is 8 bits in length and specifies which of a number of general alarms MAY be generated, as  
 3473 listed in Table 3-14. A ‘1’ in each bit position enables the associated alarm.

<sup>10</sup> CCB 2197 new values for Atrium and Mirror because of conflict with ZSE for value 0x01

3474

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

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

3475

3476 These alarms are provided as basic alarms that a device MAY use even if no other clusters with alarms are present on  
3477 the device.

### 3478 **3.2.2.2.20 DisableLocalConfig Attribute**

3479 The *DisableLocalConfig* attribute allows a number of local device configuration functions to be disabled.

3480

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

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

3481 The intention of this attribute is to allow disabling of any local configuration user interface, for example to prevent  
3482 reset or binding buttons being activated by non-authorized persons in a public building.

3483 Bit 0 of the *DisableLocalConfig* attribute disables any factory reset button (or equivalent) on the device. Bit 1 disables  
3484 any device configuration button(s) (or equivalent)—for example, a bind button.

### 3485 **3.2.2.2.21 SWBuildID Attribute**

3486 The *SWBuildID* attribute represents a detailed, manufacturer-specific reference to the version of the software.

## 3487 **3.2.2.3 Commands Received**

3488 The command IDs for the Basic cluster are listed in Table 3-16.

3489

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

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

### 3490 **3.2.2.3.1 Reset to Factory Defaults Command**

3491 This command does not have a payload.

#### 3492 **3.2.2.3.1.1 Effect on Receipt**

3493 On receipt of this command, the device resets all the attributes of all its clusters to their factory defaults.

3494 Note that networking functionality, bindings, groups, or other persistent data are not affected by this command.

3495 **3.2.2.4 Commands Generated**

3496 No commands are generated by the server cluster.

3497 **3.2.3 Client**

3498 The client has no dependencies or attributes. No cluster specific commands are received by the client.

3499 The cluster specific commands generated by the client cluster are those received by the server, as required by the  
 3500 application.

3501 **3.3 Power Configuration**

---

3502 **3.3.1 Overview**

3503 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 3504 etc.

3505 Attributes for determining detailed information about a device’s power source(s) and for configuring under/over  
 3506 voltage alarms.

3507 **3.3.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added; mains power lost alarm added to <i>MainsAlarmMask</i> ; CCB 1809

3508 **3.3.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	PC

3509 **3.3.1.3 Cluster Identifiers**

Identifier	Name
0x0001	Power Configuration

3510 **3.3.2 Server**

3511 **3.3.2.1 Dependencies**

3512 Any endpoint that implements this server cluster SHALL also implement the Basic server cluster.

3513 For the alarm functionality described in this cluster to be operational, any endpoint that implements the Power  
 3514 Configuration server cluster must also implement the Alarms server cluster (see sub-clause Alarms).

3515 **3.3.2.2 Attributes**

3516 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 3517 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 3518 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 3519 are listed in Table 3-17.

3520 **Table 3-17. Power Configuration Attribute Sets**

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

3521 **3.3.2.2.1 Mains Information Attribute Set**

3522 The Mains Information attribute set contains the attributes summarized in Table 3-18.

3523 **Table 3-18. Attributes of the Mains Information Attribute Set**

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

3524 **3.3.2.2.1.1 MainsVoltage Attribute**

3525 The *MainsVoltage* attribute is 16 bits in length and specifies the actual (measured) RMS voltage (or DC voltage in the  
 3526 case of a DC supply) currently applied to the device, measured in units of 100mV.

3527 **3.3.2.2.1.2 MainsFrequency Attribute**

3528 The *MainsFrequency* attribute is 8 bits in length and represents the frequency, in Hertz, of the mains as determined  
 3529 by the device as follows:

3530  $MainsFrequency = 0.5 \times \text{measured frequency}$

3531 Where  $2 \text{ Hz} \leq \text{measured frequency} \leq 506 \text{ Hz}$ , corresponding to a *MainsFrequency* in the range 1 to 0xfd.

3532 The maximum resolution this format allows is 2 Hz.

3533 The following special values of *MainsFrequency* apply.

3534 0x00 indicates a frequency that is too low to be measured.

3535 0xfe indicates a frequency that is too high to be measured.

3536 0xff indicates that the frequency could not be measured.

3537 In the case of a DC supply, this attribute SHALL also have the value zero.

3538 **3.3.2.2.2 Mains Settings Attribute Set**

3539 The Mains Settings attribute set contains the attributes summarized in Table 3-19.

3540 **Table 3-19. Attributes of the Mains Settings Attribute Set**

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

3541 The alarm settings in this table require the Alarms cluster to be implemented on the same device - see Dependencies.  
 3542 If the Alarms cluster is not present on the same device, they MAY be omitted.

3543 **3.3.2.2.2.1 MainsAlarmMask Attribute**

3544 The *MainsAlarmMask* attribute is 8 bits in length and specifies which mains alarms MAY be generated, as listed in  
 3545 Table 3-20. A ‘1’ in each bit position enables the alarm.

3546 **Table 3-20. Values of the MainsAlarmMask Attribute**

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

3547 **3.3.2.2.2.2 MainsVoltageMinThreshold Attribute**

3548 The *MainsVoltageMinThreshold* attribute is 16 bits in length and specifies the lower alarm threshold, measured in  
 3549 units of 100mV, for the *MainsVoltage* attribute. The value of this attribute SHALL be less than  
 3550 *MainsVoltageMaxThreshold*.

3551 If the value of *MainsVoltage* drops below the threshold specified by *MainsVoltageMinThreshold*, the device SHALL  
 3552 start a timer to expire after *MainsVoltageDwellTripPoint* seconds. If the value of this attribute increases to greater  
 3553 than or equal to *MainsVoltageMinThreshold* before the timer expires, the device SHALL stop and reset the timer. If  
 3554 the timer expires, an alarm SHALL be generated.

3555 The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x00.

3556 If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

3557 **3.3.2.2.2.3 MainsVoltageMaxThreshold Attribute**

3558 The *MainsVoltageMaxThreshold* attribute is 16 bits in length and specifies the upper alarm threshold, measured in  
 3559 units of 100mV, for the *MainsVoltage* attribute. The value of this attribute SHALL be greater than  
 3560 *MainsVoltageMinThreshold*.

3561 If the value of *MainsVoltage* rises above the threshold specified by *MainsVoltageMaxThreshold*, the device SHALL  
 3562 start a timer to expire after *MainsVoltageDwellTripPoint* seconds. If the value of this attribute drops to lower than or  
 3563 equal to *MainsVoltageMaxThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer  
 3564 expires, an alarm SHALL be generated.

3565 The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x01.

3566 If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

#### 3567 **3.3.2.2.2.4 MainsVoltageDwellTripPoint Attribute**

3568 The *MainsVoltageDwellTripPoint* attribute is 16 bits in length and specifies the length of time, in seconds that the  
 3569 value of *MainsVoltage* MAY exist beyond either of its thresholds before an alarm is generated.

3570 If this attribute takes the value 0xffff then the associated alarms SHALL NOT be generated.

### 3571 **3.3.2.2.3 Battery Information Attribute Set**

3572 The Battery Information attribute set contains the attributes summarized in Table 3-21.

3573 **Table 3-21. Attributes of the Battery Information Attribute Set**

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

3574 Manufacturers SHOULD measure the battery voltage and capacity at a consistent moment (e.g., the moment of radio  
 3575 transmission (i.e., peak demand)) in order to avoid unnecessary fluctuation in reporting the attribute, which can  
 3576 confuse users and make them call into question the quality of the device.

3577 Manufacturers SHOULD employ a hysteresis algorithm appropriate for their battery type in order to smooth battery  
 3578 reading fluctuations and avoid sending multiple battery warning messages when crossing the voltage thresholds  
 3579 defined for warnings.

#### 3580 **3.3.2.2.3.1 BatteryVoltage Attribute**

3581 The *BatteryVoltage* attribute is 8 bits in length and specifies the current actual (measured) battery voltage, in units of  
 3582 100mV.

3583 The value 0xff indicates an invalid or unknown reading.

#### 3584 **3.3.2.2.3.2 BatteryPercentageRemaining Attribute**

3585 Specifies the remaining battery life as a half integer percentage of the full battery capacity (e.g., 34.5%, 45%, 68.5%,  
 3586 90%) with a range between zero and 100%, with 0x00 = 0%, 0x64 = 50%, and 0xC8 = 100%. This is particularly  
 3587 suited for devices with rechargeable batteries.

3588 The value 0xff indicates an invalid or unknown reading.

3589 This attribute SHALL be configurable for attribute reporting.



3590 **3.3.2.2.4 Battery Settings Attribute Set**

3591 **Table 3-22. Attributes of the Battery Settings Attribute Set**

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

3592 **3.3.2.2.4.1 BatteryManufacturer Attribute**

3593 The *BatteryManufacturer* attribute is a maximum of 16 bytes in length and specifies the name of the battery  
 3594 manufacturer as a character string.

3595 **3.3.2.2.4.2 BatterySize Attribute**

3596 The *BatterySize* attribute is an enumeration which specifies the type of battery being used by the device. This attribute  
 3597 SHALL be set to one of the non-reserved values listed in Table 3-23.

3598 **Table 3-23. Values of the *BatterySize* Attribute**

<b>Attribute Value</b>	<b>Description</b>
0x00	No battery
0x01	Built in
0x02	Other

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

3599 **3.3.2.2.4.3 BatteryAHRRating Attribute**

3600 The *BatteryAHRRating* attribute is 16 bits in length and specifies the Ampere-hour rating of the battery, measured in  
3601 units of 10mAHr.

3602 **3.3.2.2.4.4 BatteryQuantity Attribute**

3603 The *BatteryQuantity* attribute is 8 bits in length and specifies the number of battery cells used to power the device.

3604 **3.3.2.2.4.5 BatteryRatedVoltage Attribute**

3605 The *BatteryRatedVoltage* attribute is 8 bits in length and specifies the rated voltage of the battery being used in the  
3606 device, measured in units of 100mV.

3607 **3.3.2.2.4.6 BatteryAlarmMask Attribute**

3608 The *BatteryAlarmMask* attribute specifies which battery alarms must be generated, as listed in Table 3-24. A ‘1’ in  
3609 each bit position enables the alarm.

3610 **Table 3-24. Values of the *BatteryAlarmMask* Attribute**

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

3611 Manufacturers are responsible for determining the capability to sense and levels at which the alarms are generated.  
3612 See Section 10.3.2, References, for additional recommendations on measuring battery voltage.

3613 **3.3.2.2.4.7 BatteryVoltageMinThreshold Attribute**

- 3614 Specifies the low battery voltage alarm threshold, measured in units of 100mV at which the device can no longer  
 3615 operate or transmit via its radio (i.e., last gasp).
- 3616 If the value of *BatteryVoltage* drops below the threshold specified by *BatteryVoltageMinThreshold*, an appropriate  
 3617 alarm SHALL be generated and/or the corresponding bit SHALL be updated in the *BatteryAlarmState* attribute.
- 3618 In order to report to Power Configuration clients, servers that implement *BatteryVoltageMinThreshold* attribute  
 3619 SHALL implement alarming via the Alarm Cluster, attribute reporting via the *BatteryAlarmState* attribute, or both.
- 3620 For servers that implement alarming via the Alarm Cluster, the appropriate alarm is specified in the Alarm Code field  
 3621 (see 3.11.2.3.1) included in the generated alarm and SHALL be one of the values in Table 3-25. The host determines  
 3622 which alarm code to populate based on the *BatteryAlarmMask* attribute and the *BatteryVoltageMinThreshold* attribute  
 3623 reached. For example, when the *BatteryVoltage* attribute reaches the value specified by the  
 3624 *BatteryVoltageMinThreshold* attribute, an alarm with the Alarm Code Field Enumeration “0x10” SHALL be  
 3625 generated.
- 3626 For servers that implement battery alarm reporting via the *BatteryAlarmState* attribute, the bit corresponding to the  
 3627 threshold level reached SHALL be set to TRUE. See the *BatteryAlarmState* attribute details for more information.

3628 **Table 3-25. Alarm Code Field Enumerations for Battery Alarms**

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

3629 **3.3.2.2.4.8 BatteryVoltageThreshold 1-3 Attributes**

- 3630 Specify the low voltage alarm thresholds, measured in units of 100mV, for the *BatteryVoltage* attribute.
- 3631 If the value of *BatteryVoltage* drops below the threshold specified by a *BatteryVoltageThreshold*, an appropriate alarm  
 3632 SHALL be generated and/or the corresponding bit SHALL be updated in the *BatteryAlarmState* attribute.

3633 The *BatteryVoltageThreshold1-3* attributes SHALL be ordered in ascending order such that the *BatteryVoltage* level  
3634 specified to trigger:

- 3635 • *BatteryVoltageThreshold3* is higher than the level specified to trigger *BatteryVoltageThreshold2*
- 3636 • *BatteryVoltageThreshold2* is higher than the level specified to trigger *BatteryVoltageThreshold*
- 3637 • *BatteryVoltageThreshold1* is higher than the level specified to trigger *BatteryVoltageMinThreshold*

3638 The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL  
3639 be one of the values in Table 3-25. The host determines which alarm code to populate based on the *BatteryAlarmMask*  
3640 attribute and the *BatteryVoltageThreshold1-3* attribute reached.

3641 If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

#### 3642 **3.3.2.2.4.9 BatteryPercentageMinThreshold Attribute**

3643 Specifies the low battery percentage alarm threshold, measured in percentage (i.e., zero to 100%), for the  
3644 *BatteryPercentageRemaining* attribute (see sub-clause 3.3.2.2.3.2).

3645 If the value of *BatteryPercentageRemaining* drops below the threshold specified by a *BatteryPercentageThreshold*,  
3646 an appropriate alarm SHALL be generated.

3647 The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL  
3648 be the value in Table 3-25 that corresponds with this threshold being reached for a given battery source. The host  
3649 determines which alarm code to populate based on the *BatteryAlarmMask* attribute.

3650 If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

#### 3651 **3.3.2.2.4.10 BatteryPercentageThreshold 1-3 Attributes**

3652 Specify the low battery percentage alarm thresholds, measured in percentage (i.e., zero to 100%), for the  
3653 *BatteryPercentageRemaining* attribute (see sub-clause 3.3.2.2.3.2).

3654 If the value of *BatteryPercentageRemaining* drops below the threshold specified by a *BatteryPercentageThreshold*,  
3655 an appropriate alarm SHALL be generated.

3656 The *BatteryPercentageThreshold1-3* attributes SHALL be ordered in ascending order such that the  
3657 *BatteryPercentageRemaining* level specified to trigger:

- 3658 • *BatteryPercentageThreshold3* is higher than the level specified to trigger *BatteryPercentageThreshold2*
- 3659 • *BatteryPercentageThreshold2* is higher than the level specified to trigger *BatteryPercentageThreshold*
- 3660 • *BatteryPercentageThreshold1* is higher than the level specified to trigger *BatteryPercentageMinThreshold*

3661 The appropriate alarm is specified in the Alarm Code field (see 3.11.2.3.1) included in the generated alarm and SHALL  
3662 be one of the values in Table 3-25. The host determines which alarm code to populate based on the *BatteryAlarmMask*  
3663 attribute and the *BatteryPercentageThreshold1-3* attribute reached.

3664 If this attribute takes the value 0xff then this alarm SHALL NOT be generated.

#### 3665 **3.3.2.2.4.11 BatteryAlarmState Attribute**

3666 Specifies the current state of the device's battery alarms. This attribute provides a persistent record of a device's battery  
3667 alarm conditions as well as a mechanism for reporting changes to those conditions, including the elimination of battery  
3668 alarm states (e.g., when a battery is replaced).

3669 If implemented, the server SHALL support attribute reporting for *BatteryAlarmState* attribute. This provides clients  
3670 with a mechanism for reading the current state in case they missed the initial attribute report and also reduces network  
3671 and battery use due to repeated polling of this attribute when it has not changed. It also provides a way of notifying  
3672 clients when battery alarm conditions no longer exist (e.g., when the batteries have been replaced).

3673

**Table 3-26. BatteryAlarmState Enumerations**

<b>Bit</b>	<b>Description</b>
0	<i>BatteryVoltageMinThreshold</i> or <i>BatteryPercentageMinThreshold</i> reached for Battery Source 1
1	<i>BatteryVoltageThreshold1</i> or <i>BatteryPercentageThreshold1</i> reached for Battery Source 1
2	<i>BatteryVoltageThreshold2</i> or <i>BatteryPercentageThreshold2</i> reached for Battery Source 1
3	<i>BatteryVoltageThreshold3</i> or <i>BatteryPercentageThreshold3</i> reached for Battery Source 1
10	<i>BatteryVoltageMinThreshold</i> or <i>BatteryPercentageMinThreshold</i> reached for Battery Source 2
11	<i>BatteryVoltageThreshold1</i> or <i>BatteryPercentageThreshold1</i> reached for Battery Source 2
12	<i>BatteryVoltageThreshold2</i> or <i>BatteryPercentageThreshold2</i> reached Battery Source 2
13	<i>BatteryVoltageThreshold3</i> or <i>BatteryPercentageThreshold3</i> reached Battery Source 2
20	<i>BatteryVoltageMinThreshold</i> or <i>BatteryPercentageMinThreshold</i> reached for Battery Source 3
21	<i>BatteryVoltageThreshold1</i> or <i>BatteryPercentageThreshold1</i> reached for Battery Source 3
22	<i>BatteryVoltageThreshold2</i> or <i>BatteryPercentageThreshold2</i> reached Battery Source 3
23	<i>BatteryVoltageThreshold3</i> or <i>BatteryPercentageThreshold3</i> reached Battery Source 3
30	Mains power supply lost/unavailable (i.e., device is running on battery)

3674 Manufacturers are responsible for determining the capability to sense and levels at which the alarms are generated.  
 3675 See 3.3.2.2.3 for additional recommendations on measuring battery voltage.

3676 **3.3.2.2.5 Battery Information 2 Attribute Set**

3677 This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery  
 3678 Information Attribute Set and provides a host with the ability to represent battery information for a secondary battery  
 3679 bank or cell.

3680 **3.3.2.2.6 Battery Settings 2 Attribute Set**

3681 This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery  
 3682 Settings Attribute Set and provides a host with the ability to represent battery settings for a secondary battery bank or  
 3683 cell.

3684 **3.3.2.2.7 Battery Information 3 Attribute Set**

3685 This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery  
 3686 Information Attribute Set and provides a host with the ability to represent battery information for a tertiary battery  
 3687 bank or cell.

**3688 3.3.2.2.8 Battery Settings 3 Attribute Set**

3689 This attribute set is an exact replica of all the attributes, commands, and behaviors contained within the Battery  
3690 Settings Attribute Set and provides a host with the ability to represent battery settings for a tertiary battery bank or  
3691 cell. Commands Received

**3692 3.3.2.3 Commands Received**

3693 No commands are received by the server.

**3694 3.3.2.4 Commands Generated**

3695 The server generates no commands.

**3696 3.3.3 Client**

3697 The client has no dependencies or cluster specific attributes. There are no cluster specific commands that are generated  
3698 or received by the client

**3699 3.4 Device Temperature Configuration**

---

**3700 3.4.1 Overview**

3701 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
3702 etc.

3703 Attributes for determining information about a device's internal temperature, and for configuring under/over  
3704 temperature alarms for temperatures that are outside the device's operating range.

**3705 3.4.1.1 Revision History**

Rev	Description
1	global mandatory <i>Cluster Revision</i> attribute added

**3706 3.4.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	DTMP

**3707 3.4.1.3 Cluster Identifiers**

Identifier	Name
0x0002	Device Temperature Configuration

3708 **3.4.2 Server**

3709 **3.4.2.1 Dependencies**

3710 For the alarm functionality described in this cluster to be operational, any endpoint that implements the Device  
 3711 Temperature Configuration server cluster SHALL also implement the Alarms server cluster (see 3.11).

3712 **3.4.2.2 Attributes**

3713 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 3714 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 3715 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 3716 are listed in Table 3-27.

3717 **Table 3-27. Device Temperature Configuration Attribute Sets**

Attribute Set Identifier	Description
0x000	Device Temperature Information
0x001	Device Temperature Settings

3718 **3.4.2.2.1 Device Temperature Information Attribute Set**

3719 The Device Temperature Information attribute set contains the attributes summarized in Table 3-28.

3720 **Table 3-28. Device Temperature Information Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0000	<i>CurrentTemperature</i>	int16	-200 to +200	R	-	M
0x0001	<i>MinTempExperienced</i>	int16	-200 to +200	R	-	O
0x0002	<i>MaxTempExperienced</i>	int16	-200 to +200	R	-	O
0x0003	<i>OverTempTotalDwell</i>	uint16	0x0000 to 0xffff	R	0	O

3721 **3.4.2.2.1.1 CurrentTemperature Attribute**

3722 The *CurrentTemperature* attribute is 16 bits in length and specifies the current internal temperature, in degrees Celsius,  
 3723 of the device. This attribute SHALL be specified in the range –200 to +200.

3724 The value 0xffff indicates an invalid reading.

3725 **3.4.2.2.1.2 MinTempExperienced Attribute**

3726 The *MinTempExperienced* attribute is 16 bits in length and specifies the minimum internal temperature, in degrees  
 3727 Celsius, the device has experienced while powered. This attribute SHALL be specified in the range –200 to +200.

3728 The value 0xffff indicates an invalid reading.

3729 **3.4.2.2.1.3 MaxTempExperienced Attribute**

3730 The *MaxTempExperienced* attribute is 16 bits in length and specifies the maximum internal temperature, in degrees  
3731 Celsius, the device has experienced while powered. This attribute SHALL be specified in the range –200 to +200.

3732 The value 0xffff indicates an invalid reading.

3733 **3.4.2.2.1.4 OverTempTotalDwell Attribute**

3734 The *OverTempTotalDwell* attribute is 16 bits in length and specifies the length of time, in hours; the device has spent  
3735 above the temperature specified by the *HighTempThreshold* attribute 3.4.2.2.2.3, cumulative over the lifetime of the  
3736 device.

3737 The value 0xffff indicates an invalid time.

3738 **3.4.2.2.2 Device Temperature Settings Attribute Set**

3739 The Device Temperature Settings attribute set contains the attributes summarized in Table 3-29.

3740 **Table 3-29. Device Temperature Settings Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0010	<i>DeviceTempAlarmMask</i>	map8	0000 00xx	RW	0000 0000	O
0x0011	<i>LowTempThreshold</i>	int16	-200 to +200	RW	-	O
0x0012	<i>HighTempThreshold</i>	int16	-200 to +200	RW	-	O
0x0013	<i>LowTempDwellTripPoint</i>	uint24	0x000000 to 0xffffff	RW	-	O
0x0014	<i>HighTempDwellTripPoint</i>	uint24	0x000000 to 0xffffff	RW	-	O

3741 All attributes in this table require the Alarms cluster to be implemented on the same device - see Dependencies. If the  
3742 Alarms cluster is not present on the same device, they MAY be omitted.

3743 **3.4.2.2.2.1 DeviceTempAlarmMask Attribute**

3744 The *DeviceTempAlarmMask* attribute is 8 bits in length and specifies which alarms MAY be generated, as listed in  
3745 Table 3-30. A ‘1’ in each bit position enables the corresponding alarm.

3746 **Table 3-30. Values of the DeviceTempAlarmMask Attribute**

Attribute Bit Number	Alarm
0	Device Temperature too low
1	Device Temperature too high

3747 **3.4.2.2.2.2 LowTempThreshold Attribute**

3748 The *LowTempThreshold* attribute is 16 bits in length and specifies the lower alarm threshold, measured in degrees  
3749 Celsius (range -200°C to 200°C), for the *CurrentTemperature* attribute. The value of this attribute SHALL be less  
3750 than *HighTempThreshold*.



3751 If the value of *CurrentTemperature* drops below the threshold specified by *LowTempThreshold*, the device SHALL  
3752 start a timer to expire after *LowTempDwellTripPoint* seconds. If the value of this attribute increases to greater than or  
3753 equal to *LowTempThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer expires,  
3754 an alarm SHALL be generated.

3755 The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x00.

3756 If this attribute takes the value 0x8000 then this alarm SHALL NOT be generated.

#### 3757 **3.4.2.2.2.3 HighTempThreshold Attribute**

3758 The *HighTempThreshold* attribute is 16 bits in length and specifies the upper alarm threshold, measured in degrees  
3759 Celsius (range -200°C to 200°C), for the *CurrentTemperature* attribute. The value of this attribute SHALL be greater  
3760 than *LowTempThreshold*.

3761 If the value of *CurrentTemperature* rises above the threshold specified by *HighTempThreshold*, the device SHALL  
3762 start a timer to expire after *HighTempDwellTripPoint* seconds. If the value of this attribute drops to lower than or  
3763 equal to *HighTempThreshold* before the timer expires, the device SHALL stop and reset the timer. If the timer expires,  
3764 an alarm SHALL be generated.

3765 The Alarm Code field (see 3.11.2.4.1) included in the generated alarm SHALL be 0x01.

3766 If this attribute takes the value 0x8000 then this alarm SHALL NOT be generated.

#### 3767 **3.4.2.2.2.4 LowTempDwellTripPoint Attribute**

3768 The *LowTempDwellTripPoint* attribute is 24 bits in length and specifies the length of time, in seconds, that the value  
3769 of *CurrentTemperature* MAY exist below *LowTempThreshold* before an alarm is generated.

3770 If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

#### 3771 **3.4.2.2.2.5 HighTempDwellTripPoint Attribute**

3772 The *HighTempDwellTripPoint* attribute is 24 bits in length and specifies the length of time, in seconds, that the value  
3773 of *CurrentTemperature* MAY exist above *HighTempThreshold* before an alarm is generated.

3774 If this attribute takes the value 0xffff then this alarm SHALL NOT be generated.

### 3775 **3.4.2.3 Commands**

3776 No commands are received or generated by the server.

## 3777 **3.4.3 Client**

3778 The client has no dependencies or cluster specific attributes. There are no cluster specific commands that are generated  
3779 or received by the client.

## 3780 **3.5 Identify**

---

### 3781 **3.5.1 Overview**

3782 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
3783 etc.

3784 Attributes and commands to put a device into an Identification mode (e.g., flashing a light), that indicates to an  
3785 observer – e.g., an installer - which of several devices it is, also to request any device that is identifying itself to  
3786 respond to the initiator.

3787 Note that this cluster cannot be disabled, and remains functional regardless of the setting of the *DeviceEnable* attribute  
3788 in the Basic cluster.

### 3789 3.5.1.1 Revision History

Rev	Description
1	global mandatory <i>Cluster Revision</i> attribute added

### 3790 3.5.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	I

### 3791 3.5.1.3 Cluster Identifiers

Identifier	Name
0x0003	Identify

## 3792 3.5.2 Server

### 3793 3.5.2.1 Attributes

3794 The server supports the attribute shown in Table 3-31.

3795 **Table 3-31. Attributes of the Identify Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>IdentifyTime</i>	uint16	0x0000 to 0xffff	RW	0x0000	M

#### 3796 3.5.2.1.1 *IdentifyTime* Attribute

3797 The *IdentifyTime* attribute specifies the remaining length of time, in seconds, that the device will continue to identify  
3798 itself.

3799 If this attribute is set to a value other than 0x0000 then the device SHALL enter its identification procedure, in order  
3800 to indicate to an observer which of several devices it is. It is recommended that this procedure consists of flashing a  
3801 light with a period of 0.5 seconds. The *IdentifyTime* attribute SHALL be decremented every second.

3802 If this attribute reaches or is set to the value 0x0000 then the device SHALL terminate its identification procedure.

### 3803 3.5.2.2 Commands Received

3804 The server side of the identify cluster is capable of receiving the commands listed in Table 3-32.

3805 **Table 3-32. Received Command IDs for the Identify Cluster**

Command Identifier	Description	M/O
0x00	Identify	M
0x01	Identify Query	M
0x40	Trigger effect	O

3806 **3.5.2.2.1 Identify Command**

3807 The identify command starts or stops the receiving device identifying itself.

3808 **3.5.2.2.1.1 Payload Format**

3809 The identify query response command payload SHALL be formatted as illustrated in Figure 3-7.

3810 **Figure 3-7. Format of Identify Query Response Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Identify Time

3811 **3.5.2.2.1.2 Effect on Receipt**

3812 On receipt of this command, the device SHALL set the *IdentifyTime* attribute to the value of the Identify Time field.  
 3813 This then starts, continues, or stops the device's identification procedure as detailed in 3.5.2.1.1.

3814 **3.5.2.2.2 Identify Query Command**

3815 The identify query command allows the sending device to request the target or targets to respond if they are currently  
 3816 identifying themselves.

3817 This command has no payload.

3818 **3.5.2.2.2.1 Effect on Receipt**

3819 On receipt of this command, if the device is currently identifying itself then it SHALL generate an appropriate Identify  
 3820 Query Response command, see 3.5.2.4.1 and unicast it to the requestor. If the device is not currently identifying itself  
 3821 it SHALL take no further action.

3822 **3.5.2.2.3 Trigger Effect Command**

3823 The *Trigger Effect* command allows the support of feedback to the user, such as a certain light effect. It is used to  
 3824 allow an implementation to provide visual feedback to the user under certain circumstances such as a color light  
 3825 turning green when it has successfully connected to a network. The use of this command and the effects themselves  
 3826 are entirely up to the implementer to use whenever a visual feedback is useful but it is not the same as and does not  
 3827 replace the identify mechanism used during commissioning.

3828 The payload of this command SHALL be formatted as illustrated in Figure 3-8.

3829

**Figure 3-8. Format of the Trigger Effect Command**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	Effect identifier	Effect variant

3830 **3.5.2.2.3.1 Effect Identifier Field**

3831 The *Effect Identifier* field is 8-bits in length and specifies the identify effect to use. This field SHALL contain one of  
3832 the nonreserved values listed in Table 3-33.

3833 **Table 3-33. Values of the Effect Identifier Field of the Trigger Effect Command**

<b>Effect Identifier Field Value</b>	<b>Effect<sup>11</sup></b>	<b>Notes</b>
0x00	Blink	e.g., Light is turned on/off once.
0x01	Breathe	e.g., Light turned on/off over 1 second and repeated 15 times.
0x02	Okay	e.g., Colored light turns green for 1 second; noncolored light flashes twice.
0x0b	Channel change	e.g., Colored light turns orange for 8 seconds; noncolored light switches to maximum brightness for 0.5s and then minimum brightness for 7.5s.
0xfe	Finish effect	Complete the current effect sequence before terminating. e.g., if in the middle of a breathe effect (as above), first complete the current 1s breathe effect and then terminate the effect.
0xff	Stop effect	Terminate the effect as soon as possible.

3834 **3.5.2.2.3.2 Effect Variant Field**

3835 The *effect variant* field is 8-bits in length and is used to indicate which variant of the effect, indicated in the *effect*  
3836 *identifier* field, SHOULD be triggered. If a device does not support the given variant, it SHALL use the default variant.  
3837 This field SHALL contain one of the non-reserved values listed in Table 3-34.

3838 **Table 3-34. Values of the Effect Variant Field of the Trigger Effect Command**

<b>Effect Variant Field Value</b>	<b>Description</b>
0x00	Default

3839 **3.5.2.2.3.3 Effect on Receipt**

3840 On receipt of this command, the device SHALL execute the trigger effect indicated in the *Effect Identifier* and *Effect*  
3841 *Variant* fields. If the *Effect Variant* field specifies a variant that is not supported on the device, it SHALL execute the  
3842 default variant.

<sup>11</sup> Implementers SHOULD indicate during testing how they handle each effect.

3843 **3.5.2.3 Commands Generated**

3844 The server side of the identify cluster is capable of generating the commands listed in Table 3-35.

3845 **Table 3-35. Generated Command IDs for the Identify Cluster**

Command Identifier Field Value	Description	M/O
0x00	Identify Query Response	M

3846 **3.5.2.3.1 Identify Query Response Command**

3847 The identify query response command is generated in response to receiving an Identify Query command, see 3.5.2.2.2,  
 3848 in the case that the device is currently identifying itself.

3849 **3.5.2.3.1.1 Payload Format**

3850 The identify query response command payload SHALL be formatted as illustrated in Figure 3-9.

3851 **Figure 3-9. Format of Identify Query Response Command Payload**

Octets	2
Data Type	uint16
Field Name	Timeout

3852 **3.5.2.3.1.2 Timeout Field**

3853 The Timeout field contains the current value of the *IdentifyTime* attribute, and specifies the length of time, in seconds,  
 3854 that the device will continue to identify itself.

3855 **3.5.2.3.1.3 Effect on Receipt**

3856 On receipt of this command, the device is informed of a device in the network which is currently identifying itself.  
 3857 This information MAY be particularly beneficial in situations where there is no commissioning tool. Note that there  
 3858 MAY be multiple responses.

3859 **3.5.3 Client**

3860 The client has no cluster specific attributes. The client generates the cluster specific commands detailed in 3.5.2.2, as  
 3861 required by the application. The client receives the cluster specific response commands detailed in 3.5.2.3.

3862 **3.6 Groups**

---

3863 **3.6.1 Overview**

3864 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 3865 etc.

3866 The stack specification provides the capability for group addressing. That is, any endpoint on any device MAY be  
 3867 assigned to one or more groups, each labeled with a 16-bit identifier (0x0001 to 0xffff), which acts for all intents and  
 3868 purposes like a network address. Once a group is established, frames, sent using the APSDE-DATA.request primitive  
 3869 and having a DstAddrMode of 0x01, denoting group addressing, will be delivered to every endpoint assigned to the  
 3870 group address named in the DstAddr parameter of the outgoing APSDE-DATA.request primitive on every device in  
 3871 the network for which there are such endpoints.

3872 Management of group membership on each device and endpoint is implemented by the APS, but the over-the-air  
 3873 messages that allow for remote management and commissioning of groups are defined here in the cluster library on  
 3874 the theory that, while the basic group addressing facilities are integral to the operation of the stack, not every device  
 3875 will need or want to implement this management cluster. Furthermore, the placement of the management commands  
 3876 here allows developers of proprietary profiles to avoid implementing the library cluster but still exploit group  
 3877 addressing.

3878 Commands are defined here for discovering the group membership of a device, adding a group, removing a group and  
 3879 removing all groups.

3880 Finally, the group cluster allows application entities to store a name string for each group to which they are assigned  
 3881 and to report that name string in response to a client request.

3882 Note that configuration of group addresses for outgoing commands is achieved using the APS binding mechanisms,  
 3883 and is not part of this cluster.

3884 As Groupcasts are made on a broadcast to all devices for which macRxOnWhenIdle = TRUE, sleeping end devices  
 3885 will not be able to benefit from the features of the Groups and Scenes server Cluster. For example, a door lock which  
 3886 would typically be a sleeping end device would not be able to receive the datagrams required to commission a scene  
 3887 or change for example, to a night scene. It is therefore not Mandatory but only optional to support the Groups and  
 3888 Scenes Server cluster if the device is a Sleeping end device (even when listed as Mandatory).

### 3889 3.6.1.1 Revision History

Rev	Description
1	global mandatory <i>Cluster Revision</i> attribute added; CCB 1745 2100
2	CCB 2289

### 3890 3.6.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	G

### 3891 3.6.1.3 Cluster Identifiers

Identifier	Name
0x0004	Groups

3892 <sup>12</sup>

<sup>12</sup> CCB 2289 removed Security clause about Permissions Configuration Table

3893 **3.6.2 Server**

3894 Each device that implements this cluster MAY be thought of as a group management server in the sense that it responds  
 3895 to information requests and configuration commands regarding the contents of its group table.

3896 Note that, since these commands are simply data frames sent using the APSDE\_SAP, they must be addressed with  
 3897 respect to device and endpoint. In particular, the destination device and endpoint of a group management command  
 3898 must be unambiguous at the time of the issuance of the primitive either because:

- 3899 1. They are explicitly spelled out in the DstAddr and DstEndpoint parameters of the primitive.
- 3900 2. They are not explicitly spelled out but MAY be derived from the binding table in the APS of the sending  
 3901 device.
- 3902 3. Broadcast addressing is being employed, either with respect to the device address or the endpoint identifier.
- 3903 4. Group addressing is being employed.

3904 On receipt of a group cluster command, the APS will, at least conceptually, deliver the frame to each destination  
 3905 endpoint spelled out in the addressing portion of the APS header and, again conceptually speaking, the application  
 3906 entity resident at that endpoint will process the command and respond as necessary. From an implementation  
 3907 standpoint, of course, this MAY be done in a more economical way that does not involve duplication and separate  
 3908 processing, e.g., by providing a hook in the APS whereby group cluster commands could be delivered to a special  
 3909 application entity without duplication.

3910 **3.6.2.1 Dependencies**

3911 For correct operation of the 'Add group if identifying' command, any endpoint that implements the Groups server  
 3912 cluster SHALL also implement the Identify server cluster.

3913 **3.6.2.2 Attributes**

3914 The server supports the attribute shown in Table 3-36.

3915 **Table 3-36. Attributes of the Groups Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>NameSupport</i>	map8	x0000000	R	-	M

3916 **3.6.2.2.1 NameSupport Attribute**

3917 The most significant bit of the *NameSupport* attribute indicates whether or not group names are supported. A value of  
 3918 1 indicates that they are supported, and a value of 0 indicates that they are not supported.

3919 **3.6.2.2.2 Group Names**

3920 Group names are between 0 and 16 characters long. Support of group names is optional, and is indicated by the  
 3921 *NameSupport* attribute. Group names, if supported, must be stored in a separate data structure managed by the  
 3922 application in which the entries correspond to group table entries.

3923 **3.6.2.3 Commands Received**

3924 The groups cluster is concerned with management of the group table on a device. In practice, the group table is  
 3925 managed by the APS and the table itself is available to the next higher layer as an AIB attribute. A command set is  
 3926 defined here and the implementation details of that command set in terms of the facilities provided by the APS is left  
 3927 up to the implementer of the cluster library itself.

3928 The server side of the groups cluster is capable of receiving the commands listed in Table 3-37.

3929 **Table 3-37. Received Command IDs for the Groups Cluster**

Command Identifier Field Value	Description	M/O
0x00	Add group	M
0x01	View group	M
0x02	Get group membership	M
0x03	Remove group	M
0x04	Remove all groups	M
0x05	Add group if identifying	M

3930 **3.6.2.3.1 Generic Usage Notes**

3931 On receipt of the *Add Group*, *View Group*, or *Remove Group* command frames via the groupcast or broadcast  
 3932 transmission service, no response SHALL be given.

3933 **3.6.2.3.2 Add Group Command**

3934 The Add Group command allows the sending device to add group membership in a particular group for one or more  
 3935 endpoints on the receiving device.

3936 **3.6.2.3.2.1 Payload Format**

3937 The Add Group command payload SHALL be formatted as illustrated in Figure 3-10.

3938 **Figure 3-10. Format of the Add Group Command Payload**

Octets	2	Variable
Data Type	uint16	string
Field Name	Group ID	Group Name

3939 **3.6.2.3.2.2 Effect on Receipt**



3940 On receipt of this command, the device SHALL (if possible) add the Group ID and Group Name to its Group Table.  
 3941 If Group Name is not supported, the Group Name field SHALL be ignored. Except for the restrictions listed in  
 3942 3.6.2.3.1, the device SHALL then generate an appropriate Add Group Response command indicating success or  
 3943 failure. See 3.6.2.4.1.

3944 **3.6.2.3.3 View Group Command**

3945 The view group command allows the sending device to request that the receiving entity or entities respond with a view  
 3946 group response command containing the application name string for a particular group.

3947 **3.6.2.3.3.1 Payload Format**

3948 The View Group command payload SHALL be formatted as illustrated in Figure 3-11:

3949 **Figure 3-11. Format of the View Group Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Group ID

3950 **3.6.2.3.3.2 Effect on Receipt**

3951 Except for the restrictions listed in 3.6.2.3.1, the device, on receipt of this command, SHALL generate an appropriate  
 3952 View Group Response command 3.6.2.4.2.

3953 **3.6.2.3.4 Get Group Membership Command**

3954 The get group membership command allows the sending device to inquire about the group membership of the  
 3955 receiving device and endpoint in a number of ways.

3956 **3.6.2.3.4.1 Payload Format**

3957 The get group membership command payload SHALL be formatted as illustrated in Figure 3-12.

3958 **Figure 3-12. Format of Get Group Membership Command Payload**

<b>Octets</b>	1	Variable
<b>Data Type</b>	uint8	List of 16-bit integers
<b>Field Name</b>	Group count	Group list

3959 **3.6.2.3.4.2 Effect on Receipt**

3960 On receipt of the get group membership command, each receiving entity SHALL respond with group membership  
 3961 information using the get group membership response frame as follows:

3962 If the group count field of the command frame has a value of 0 indicating that the group list field is empty, the entity  
 3963 SHALL respond with all group identifiers of which the entity is a member.

3964 If the group list field of the command frame contains at least one group of which the entity is a member, the entity  
 3965 SHALL respond with each entity group identifier that match a group in the group list field.

3966 If the group count is non-zero, and the group list field of the command frame does not contain any group of which the  
3967 entity is a member, the entity SHALL only respond if the command is unicast. The response SHALL return a group  
3968 count of zero.

### 3969 **3.6.2.3.5 Remove Group Command**

3970 The remove group command allows the sender to request that the receiving entity or entities remove their membership,  
3971 if any, in a particular group.

3972 Note that if a group is removed the scenes associated with that group SHOULD be removed.

#### 3973 **3.6.2.3.5.1 Payload Format**

3974 The Remove Group command payload SHALL be formatted as illustrated in Figure 3-13.

3975 **Figure 3-13. Format of the Remove Group Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Group ID

#### 3976 **3.6.2.3.5.2 Effect on Receipt**

3977 On receipt of this command, the device SHALL (if possible) remove the Group ID and Group Name from its Group  
3978 Table. Except for the restrictions listed in 3.6.2.3.1, the device SHALL then generate an appropriate Remove Group  
3979 Response command indicating success or failure. See 3.6.2.4.4.

### 3980 **3.6.2.3.6 Remove All Groups Command**

3981 The remove all groups command allows the sending device to direct the receiving entity or entities to remove all group  
3982 associations.

3983 Note that removing all groups necessitates the removal of all associated scenes as well. (Note: scenes not associated  
3984 with a group need not be removed).

#### 3985 **3.6.2.3.6.1 Payload Format**

3986 The Remove All Groups command has no payload.

#### 3987 **3.6.2.3.6.2 Effect on Receipt**

3988 On receipt of this command, the device SHALL remove all groups on this endpoint from its Group Table.

### 3989 **3.6.2.3.7 Add Group If Identifying Command**

3990 The add group if identifying command allows the sending device to add group membership in a particular group for  
3991 one or more endpoints on the receiving device, on condition that it is identifying itself. Identifying functionality is  
3992 controlled using the identify cluster, (see 3.5).

3993 This command might be used to assist configuring group membership in the absence of a commissioning tool.

#### 3994 **3.6.2.3.7.1 Payload Format**

3995 The Add Group If Identifying command payload SHALL be formatted as illustrated in Figure 3-14.

3996

**Figure 3-14. Add Group If Identifying Command Payload**

<b>Octets</b>	2	Variable
<b>Data Type</b>	uint16	string
<b>Field Name</b>	Group ID	Group Name

3997 **3.6.2.3.7.2 Effect on Receipt**

3998 On receipt of this command, the device SHALL first check whether it is currently identifying itself. If so then the  
 3999 device SHALL (if possible) add the Group ID and Group Name to its Group Table. If the device it not currently  
 4000 identifying itself then no action SHALL be taken.

4001 No response is defined as this command is EXPECTED to be multicast or broadcast.

4002 **3.6.2.4 Commands Generated**

4003 The commands generated by the server side of the group cluster, as listed in Table 3-38, are responses to the received  
 4004 commands listed in sub-clause 3.6.2.3.

4005 **Table 3-38. Generated Command IDs for the Groups Cluster**

<b>Command Identifier Field Value</b>	<b>Description</b>	<b>M/O</b>
0x00	Add group response	M
0x01	View group response	M
0x02	Get group membership response	M
0x03	Remove group response	M

4006

4007 **Note:** There is no need for a response to the Remove all Groups command, as, at an application level, this command  
 4008 always succeeds.

4009 **3.6.2.4.1 Add Group Response Command**

4010 The add group response is sent by the groups cluster server in response to an add group command.

4011 **3.6.2.4.1.1 Payload Format**

4012 The Add Group Response command payload SHALL be formatted as illustrated in Figure 3-15.

4013

**Figure 3-15. Format of the Add Group Response Command Payload**

Octets	1	2
Data Type	enum8	uint16
Field Name	Status	Group ID

4014 **3.6.2.4.1.2 When Generated**

4015 This command is generated in response to a received Add Group command. The Status field is set to SUCCESS,  
4016 DUPLICATE\_EXISTS, or INSUFFICIENT\_SPACE as appropriate. The Group ID field is set to the Group ID field  
4017 of the received Add Group command.

4018 **3.6.2.4.2 View Group Response Command**

4019 The view group response command is sent by the groups cluster server in response to a view group command.

4020 **3.6.2.4.2.1 Payload Format**

4021 The View Group Response command payload SHALL be formatted as illustrated in Figure 3-16.

4022

**Figure 3-16. Format of the View Group Response Command Payload**

Octets	1	2	Variable
Data Type	enum8	uint16	string
Field Name	Status	Group ID	Group Name

4023 **3.6.2.4.2.2 When Generated**

4024 This command is generated in response to a received View Group command. The Status field is set to SUCCESS or  
4025 NOT\_FOUND as appropriate. The Group ID field is set to the Group ID field of the received View Group command.  
4026 If the status is SUCCESS, and group names are supported, the Group Name field is set to the Group Name associated  
4027 with that Group ID in the Group Table; otherwise it is set to the null (empty) string, i.e., a single octet of value 0.

4028 **3.6.2.4.3 Get Group Membership Response Command**

4029 The get group membership response command is sent by the groups cluster server in response to a get group  
4030 membership command.

4031 **3.6.2.4.3.1 Payload Format**

4032 The payload of the get group membership response command is formatted as shown in Figure 3-17.

4033

**Figure 3-17. Format of the Get Group Membership Response Command Payload**

<b>Octets</b>	1	1	Variable
<b>Data Type</b>	uint8	uint8	List of 16-bit group ID
<b>Field Name</b>	Capacity	Group count	Group list

4034

4035 The fields of the get group membership response command have the following semantics:

4036 The Capacity field SHALL contain the remaining capacity of the group table of the device. The following values  
 4037 apply:

- 4038 0 No further groups MAY be added.
- 4039 0 < Capacity < 0xfe Capacity holds the number of groups that MAY be added
- 4040 0xfe At least 1 further group MAY be added (exact number is unknown)
- 4041 0xff It is unknown if any further groups MAY be added

4042 The Group count field SHALL contain the number of groups contained in the group list field.

4043 The Group list field SHALL contain the identifiers either of all the groups in the group table (in the case where the  
 4044 group list field of the received get group membership command was empty) or all the groups from the group list field  
 4045 of the received get group membership command which are in the group table. If the total number of groups will cause  
 4046 the maximum payload length of a frame to be exceeded, then the Group list field shall contain only as many groups  
 4047 as will fit.

4048 **3.6.2.4.3.2 When Generated**

4049 See Get Group Membership Command 3.6.2.3.4.2 Effect on Receipt.

4050 **3.6.2.4.4 Remove Group Response Command**

4051 The remove group response command is generated by an application entity in response to the receipt of a remove  
 4052 group command.

4053 **3.6.2.4.4.1 Payload Format**

4054 The Remove Group Response command payload SHALL be formatted as illustrated in Figure 3-18.

4055

**Figure 3-18. Format of Remove Group Response Command Payload**

<b>Octets</b>	1	2
<b>Data Type</b>	enum8	uint16
<b>Field Name</b>	Status	Group ID

4056 **3.6.2.4.4.2 When Generated**

4057 This command is generated in response to a received Remove Group command. The Status field is set to SUCCESS  
 4058 or NOT\_FOUND as appropriate. The Group ID field is set to the Group ID field of the received Remove Group  
 4059 command.

4060 **3.6.3 Client**

4061 The Client cluster has no cluster specific attributes. The client generates the cluster specific commands detailed in  
4062 3.6.2.3. The client receives the cluster specific response commands detailed in 3.6.2.4.

4063 **3.7 Scenes**4064 **3.7.1 Overview**

4065 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4066 etc.

4067 The scenes cluster provides attributes and commands for setting up and recalling scenes. Each scene corresponds to a  
4068 set of stored values of specified attributes for one or more clusters on the same end point as the scenes cluster.

4069 In most cases scenes are associated with a particular group ID. Scenes MAY also exist without a group, in which case  
4070 the value 0x0000 replaces the group ID. Note that extra care is required in these cases to avoid a scene ID collision,  
4071 and that commands related to scenes without a group MAY only be unicast, i.e., they MAY not be multicast or  
4072 broadcast.

4073 **3.7.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added; CCB 1745
2	Recall Scene Transition Time field

4074 **3.7.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	S	Type 1 (client to server)

4075 **3.7.1.3 Cluster Identifiers**

Identifier	Name
0x0005	Scenes

4076 **3.7.2 Server**4077 **3.7.2.1 Dependencies**

4078 Any endpoint that implements the Scenes server cluster SHALL also implement the Groups server cluster.

4079 **3.7.2.2 Attributes**

4080 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 4081 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 4082 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 4083 are listed in Table 3-39.

4084 **Table 3-39. Scenes Attribute Sets**

Attribute Set Identifier	Description
0x000	Scene Management Information

4085 **3.7.2.2.1 Scene Management Information Attribute Set**

4086 The Scene Management Information attribute set contains the attributes summarized in Table 3-40.

4087 **Table 3-40. Scene Management Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>SceneCount</i>	uint8	0x00 to 0xff (see 3.7.2.3.2)	R	0x00	M
0x0001	<i>CurrentScene</i>	uint8	0x00 to 0xff (see 3.7.2.3.2)	R	0x00	M
0x0002	<i>CurrentGroup</i>	uint16	0x0000 to 0xffff	R	0x00	M
0x0003	<i>SceneValid</i>	bool	0x00 to 0x01	R	0x00	M
0x0004	<i>NameSupport</i>	map8	x0000000	R	-	M
0x0005	<i>LastConfiguredBy</i>	EUI64	-	R	-	O

4088 **3.7.2.2.1.1 SceneCount Attribute**

4089 The *SceneCount* attribute specifies the number of scenes currently in the device's scene table.

4090 **3.7.2.2.1.2 CurrentScene Attribute**

4091 The *CurrentScene* attribute holds the Scene ID of the scene last invoked.

4092 **3.7.2.2.1.3 CurrentGroup Attribute**

4093 The *CurrentGroup* attribute holds the Group ID of the scene last invoked, or 0x0000 if the scene last invoked is not  
 4094 associated with a group.

4095 **3.7.2.2.1.4 SceneValid Attribute**

4096 The *SceneValid* attribute indicates whether the state of the device corresponds to that associated with the *CurrentScene*  
 4097 and *CurrentGroup* attributes. TRUE indicates that these attributes are valid, FALSE indicates that they are not valid.

4098 Before a scene has been stored or recalled, this attribute is set to FALSE. After a successful Store Scene or Recall  
 4099 Scene command it is set to TRUE. If, after a scene is stored or recalled, the state of the device is modified, this attribute  
 4100 is set to FALSE.

4101 **3.7.2.2.1.5 NameSupport Attribute**

4102 The most significant bit of the *NameSupport* attribute indicates whether or not scene names are supported. A value of  
4103 1 indicates that they are supported, and a value of 0 indicates that they are not supported.

#### 4104 **3.7.2.2.1.6 LastConfiguredBy Attribute**

4105 The *LastConfiguredBy* attribute is 64 bits in length and specifies the IEEE address of the device that last configured  
4106 the scene table.

4107 The value 0xffffffffffffff indicates that the device has not been configured, or that the address of the device that last  
4108 configured the scenes cluster is not known.

### 4109 **3.7.2.3 Scene Table**

4110 The scene table is used to store information for each scene capable of being invoked on a device. Each scene is defined  
4111 for a particular group.

4112 The fields of each scene table entry consist of a number of sets. The base set consists of the first four fields of Table  
4113 3-41. A set of extension fields can be added by each additional cluster implemented on a device.

4114 **Table 3-41. Fields of a Scene Table Entry**

Field	Type	Valid Range	Description
Scene group ID	uint16	0x0000 to 0xffff7	The group ID for which this scene applies, or 0x0000 if the scene is not associated with a group.
Scene ID	uint8	0x00 to 0xff (see 3.7.2.3.2)	The identifier, unique within this group, which is used to identify this scene.
Scene name	string	0 to 16 characters	The name of the scene (optional)
Scene transition time	uint16	0x0000 to 0xffff	The amount of time, in seconds, it will take for the device to change from its current state to the requested scene.
Extension field sets	Variable	Variable	See the Scene Table Extensions subsections of individual clusters. Each extension field set holds a set of values of attributes for a cluster implemented on the device. The sum of all such sets defines a scene.
TransitionTime100ms	uint8	0x00 to 0x09	Together with the scene transition time element, this allows the transition time to be specified in tenths of a second.

#### 4115 **3.7.2.3.1 Scene Names**

4116 Scene names are between 0 and 16 characters long. Support of scene names is optional, and is indicated by the  
4117 *NameSupport* attribute. If scene names are not supported, any commands that write a scene name SHALL simply  
4118 discard the name, and any command that returns a scene names SHALL return the null string.

#### 4119 **3.7.2.3.2 Maximum Number of Scenes**

4120 The number of scenes capable of being stored in the table is defined by the profile in which this cluster is used. The  
4121 default maximum, in the absence of specification by the profile, is 16.



4122 **3.7.2.4 Commands Received**

4123 The received command IDs for the Scenes cluster are listed in Table 3-42.

4124 **Table 3-42. Received Command IDs for the Scenes Cluster**

Command Identifier Field Value	Description	M/O
0x00	Add Scene	M
0x01	View Scene	M
0x02	Remove Scene	M
0x03	Remove All Scenes	M
0x04	Store Scene	M
0x05	Recall Scene	M
0x06	Get Scene Membership	M
0x40	Enhanced Add Scene	O
0x41	Enhanced View Scene	O
0x42	Copy Scene	O

4125 **3.7.2.4.1 Generic Usage Notes**

4126 Scene identifier 0x00, along with group identifier 0x0000, is reserved for the global scene used by the *OnOff* cluster.

4127 On receipt of the *Add Scene*, *View Scene*, *Remove Scene*, *Remove All Scenes*, *Store Scene*, *Enhanced Add Scene*,  
 4128 *Enhanced View Scene* or *Copy Scene* command frames via the groupcast or broadcast transmission service, no  
 4129 response SHALL be given.

4130 On receipt of the *Add Scene* command, the *Scene Transition Time* element of the scene table SHALL be updated with  
 4131 the value of the *Transition Time* field and the *TransitionTime100ms* element SHALL be set to zero.

4132 **3.7.2.4.2 Add Scene Command**

4133 **3.7.2.4.2.1 Payload Format**

4134 The payload SHALL be formatted as illustrated in Figure 3-19.

4135

**Figure 3-19. Format of the Add Scene Command Payload**

<b>Octets</b>	2	1	2	Variable	Variable
<b>Data Type</b>	uint16	uint8	uint16	string	Variable (multiple types)
<b>Field Name</b>	Group ID	Scene ID	Transition time	Scene Name	Extension field sets, one per cluster

4136 The format of each extension field set is a 16 bit field carrying the cluster ID, followed by an 8 bit length field and the  
4137 set of scene extension fields specified in the relevant cluster. The length field holds the length in octets of that extension  
4138 field set.

4139 Extension field sets =

4140 {{clusterId 1, length 1, {extension field set 1}}, {clusterId 2, length 2, {extension field set 2}} ...}.

4141 The attributes included in the extension field set for each cluster are defined in the specification for that cluster in this  
4142 document (the Cluster Library). The field set consists of values for these attributes concatenated together, in the order  
4143 given in the cluster specification, with no attribute identifiers or data type indicators.

4144 For forward compatibility, reception of this command SHALL allow for the possible future addition of other attributes  
4145 to the trailing ends of the lists given in the cluster specifications (by ignoring them). Similarly, it SHALL allow for  
4146 one or more attributes to be omitted from the trailing ends of these lists (see 3.7.2.4.7.2).

4147 It is not mandatory for a field set to be included in the command for every cluster on that endpoint that has a defined  
4148 field set. Extension field sets MAY be omitted, including the case of no field sets at all.

#### 4149 **3.7.2.4.2.2 Effect on Receipt**

4150 On receipt of this command, the device SHALL (if possible) create an entry in the Scene Table with fields copied  
4151 from the command payload. If there is already a scene in the table with the same Scene ID and Group ID, it SHALL  
4152 overwrite it, (i.e., it SHALL first remove all information included in the original scene entry).

4153 Except for the restrictions in 3.7.2.4.1, the device SHALL then generate an appropriate Add Scene Response  
4154 command indicating success or failure. See 3.7.2.5.1.

### 4155 **3.7.2.4.3 View Scene Command**

#### 4156 **3.7.2.4.3.1 Payload Format**

4157 The payload SHALL be formatted as illustrated in Figure 3-20.

4158 **Figure 3-20. Format of the View Scene Command Payload**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	Group ID	Scene ID

#### 4159 **3.7.2.4.3.2 Effect on Receipt**

4160 On receipt of this command, except for the restrictions in 3.7.2.4.1, the device SHALL generate an appropriate View  
4161 Scene Response command. See 3.7.2.5.2.

4162 **3.7.2.4.4 Remove Scene Command**

4163 **3.7.2.4.4.1 Payload Format**

4164 The Remove Scene command payload SHALL be formatted as illustrated in Figure 3-21.

4165 **Figure 3-21. Format of the Remove Scene Command Payload**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	Group ID	Scene ID

4166 **3.7.2.4.4.2 Effect on Receipt**

4167 On receipt of this command, the device SHALL (if possible) remove from its Scene Table the entry with this Scene  
 4168 ID and group ID. If the command was addressed to a single device (not a group) then it SHALL generate an appropriate  
 4169 Remove Scene Response command indicating success or failure. See 3.7.2.5.3.

4170 **3.7.2.4.5 Remove All Scenes Command**

4171 **3.7.2.4.5.1 Payload Format**

4172 The Remove All Scenes command payload SHALL be formatted as illustrated in Figure 3-22.

4173 **Figure 3-22. Format of the Remove All Scenes Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Group ID

4174 **3.7.2.4.5.2 Effect on Receipt**

4175 On receipt of this command, the device SHALL, if possible, remove from its Scene Table all entries with this Group  
 4176 ID. If the command was addressed to a single device (not to a group) it SHALL then generate an appropriate Remove  
 4177 All Scenes Response command indicating success or failure. See 3.7.2.5.3.3.

4178 **3.7.2.4.6 Store Scene Command**

4179 **3.7.2.4.6.1 Payload Format**

4180 The Store Scene command payload SHALL be formatted as illustrated in Figure 3-23.

4181

**Figure 3-23. Format of the Store Scene Command Payload**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	Group ID	Scene ID

4182 **3.7.2.4.6.2 Effect on Receipt**

4183 On receipt of this command, the device SHALL (if possible) add an entry to the Scene Table with the Scene ID and  
4184 Group ID given in the command, and all extension field sets corresponding to the current state of other clusters on the  
4185 device.

4186 If an entry already exists with the same Scene ID and Group ID, as a result of a previous Add Scene command, the  
4187 extension field sets are overwritten (i.e., completely replaced) with the current values of the attributes that are  
4188 extension fields on clusters that are on the same endpoint, but the transition time field and the scene name field are  
4189 left unaltered. If no such entry exists, the transition time field SHALL be set to 0, and the scene name field SHALL  
4190 be set to the null string.

4191 Note that, accordingly, if a scene to be stored requires a transition time field and/ or a scene name field, these must be  
4192 set up by a prior Add Scene command, e.g., with no scene extension field sets.

4193 If the Group ID field is not zero, and the device is not a member of this group, the scene will not be added.

4194 If the command was addressed to a single device (not to a group) then it SHALL generate an appropriate Store Scene  
4195 Response command indicating success or failure. See 3.7.2.5.3.6.

4196 **3.7.2.4.7 Recall Scene Command**4197 **3.7.2.4.7.1 Payload Format**

4198 The Recall Scene command payload SHALL be formatted as illustrated in Figure 3-24.

4199 **Figure 3-24. Format of the Recall Scene Command Payload**

<b>Octets</b>	2	1	0/2
<b>Data Type</b>	uint16	uint8	uint16
<b>Field Name</b>	Group ID	Scene ID	Transition Time <sup>13</sup>

4200 **3.7.2.4.7.2 Effect on Receipt**

4201 On receipt of this command, the device SHALL (if possible) locate the entry in its Scene Table with the Group ID and  
4202 Scene ID given in the command. For each other cluster on the device, it SHALL then retrieve any corresponding  
4203 extension fields from the Scene Table and set the attributes and corresponding state of the cluster accordingly.

4204 If there is no extension field set for a cluster, the state of that cluster SHALL remain unchanged. If an extension field  
4205 set omits the values of any trailing attributes, the values of these attributes SHALL remain unchanged.

<sup>13</sup> NFR Add Transition Time to Recall Scene

4206 If the Transition Time field is present in the command payload and its value is not equal to 0xFFFF, this field SHALL  
 4207 indicate the transition time in 1/10ths of a second. In all other cases (command payload field not present or value equal  
 4208 to 0xFFFF), The scene transition time field of the Scene Table entry SHALL indicate the transition time. The transition  
 4209 time determines how long the transition takes from the old cluster state to the new cluster state. It is recommended  
 4210 that, where possible (e.g., it is not possible for attributes with Boolean data type), a gradual transition SHOULD take  
 4211 place from the old to the new state over this time. However, the exact transition is manufacturer dependent.  
 4212 This command does not result in a response command.

4213 **3.7.2.4.8 Get Scene Membership Command**

4214 The Get Scene Membership command can be used to find an unused scene number within the group when no  
 4215 commissioning tool is in the network, or for a commissioning tool to get used scenes for a group on a single device or  
 4216 on all devices in the group.

4217 **3.7.2.4.8.1 Payload Format**

4218 The Get Scene Membership command payload SHALL be formatted as illustrated in Figure 3-25.

4219 **Figure 3-25. Format of Get Scene Membership Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Group ID

4220 **3.7.2.4.8.2 Effect on Receipt**

4221 On receipt of this command, the device SHALL if addressed to a single device generate an appropriate Get Scene  
 4222 Membership Response command; otherwise it SHALL only generate an appropriate Get Scene Membership Response  
 4223 command if an entry within the Scene Table corresponds to the Group ID. See 3.7.2.5.3.9.

4224 **3.7.2.4.9 Enhanced Add Scene Command**

4225 The *Enhanced Add Scene* command allows a scene to be added using a finer scene transition time than the *Add Scene*  
 4226 command.

4227 The payload of this command SHALL be formatted in the same way as the *Add Scene* command, specified in the ZCL  
 4228 *Scenes* cluster, with the following difference:

4229 The *Transition Time* field SHALL be measured in tenths of a second rather than in seconds.

4230 **3.7.2.4.9.1 Effect on Receipt**

4231 On receipt of this command, the device SHALL (if possible) create an entry in the scene table with fields copied from  
 4232 the command payload. If there is already a scene in the table with the same *Scene Identifier* and *Group Identifier*, it  
 4233 SHALL overwrite it, (i.e., it SHALL first remove all information included in the original scene entry).

4234 The *Transition Time* (measured in tenths of a second) SHALL be separated into whole seconds for the standard ZCL  
 4235 *Transition Time* field of the scene table entry and the new *TransitionTime100ms* field, as specified in this specification.

4236 If the *Enhanced Add Scene* command was received via the unicast data service, the *Scenes* cluster server SHALL then  
 4237 generate and transmit an *Enhanced Add Scene Response* command back to the originator. Otherwise, no response  
 4238 SHALL be given.

4239 **3.7.2.4.10 Enhanced View Scene Command**4240 The *Enhanced View Scene* command allows a scene to be retrieved using a finer scene transition time than the *View Scene* command.  
42414242 The payload of this command SHALL be formatted in the same way as the *View Scene* command.4243 **3.7.2.4.10.1 Effect on Receipt**4244 On receipt of this command, the device SHALL generate an appropriate *enhanced view scene response* command.4245 **3.7.2.4.11 Copy Scene Command**4246 The *Copy Scene* command allows a device to efficiently copy scenes from one group/scene identifier pair to another  
4247 group/scene identifier pair.

4248 The payload of this command SHALL be formatted as illustrated in Figure 3-26.

4249 **Figure 3-26. Format of the Copy Scene Command**

<b>Octets</b>	1	2	1	2	1
<b>Data Type</b>	uint8	uint16	uint8	uint16	uint8
<b>Field Name</b>	Mode	Group identifier from	Scene identifier from	Group identifier to	Scene identifier to

4250 **3.7.2.4.11.1 Mode Field**4251 The *mode* field is 8-bits in length and contains information of how the scene copy is to proceed. This field SHALL be  
4252 formatted as illustrated in Figure 3-27.4253 **Figure 3-27. Format of the Mode Field of the Copy Scene Command**

<b>Bits: 0</b>	<b>Bits: 1-7</b>
Copy All Scenes	Reserved

4254 The *Copy All Scenes* subfield is 1-bit in length and indicates whether all scenes are to be copied. If this value is set to  
4255 1, all scenes are to be copied and the *Scene Identifier From* and *Scene Identifier To* fields SHALL be ignored.  
4256 Otherwise this field is set to 0.4257 **3.7.2.4.11.2 Group Identifier From Field**4258 The *Group Identifier From* field is 16-bits in length and specifies the identifier of the group from which the scene is  
4259 to be copied. Together with the *Scene Identifier From* field, this field uniquely identifies the scene to copy from the  
4260 scene table.4261 **3.7.2.4.11.3 Scene Identifier From Field**4262 The *Scene Identifier From* field is 8-bits in length and specifies the identifier of the scene from which the scene is to  
4263 be copied. Together with the *Group Identifier From* field, this field uniquely identifies the scene to copy from the  
4264 scene table.4265 **3.7.2.4.11.4 Group Identifier To Field**4266 The *Group Identifier To* field is 16-bits in length and specifies the identifier of the group to which the scene is to be  
4267 copied. Together with the *Scene Identifier To* field, this field uniquely identifies the scene to copy to the scene table.

4268 **3.7.2.4.11.5 Scene Identifier To Field**

4269 The *Scene Identifier To* field is 8-bits in length and specifies the identifier of the scene to which the scene is to be  
 4270 copied. Together with the *Group Identifier To* field, this field uniquely identifies the scene to copy to the scene table.

4271 **3.7.2.4.11.6 Effect on Receipt**

4272 On receipt of the *Copy Scene* command, if the *Copy All Scenes* sub-field of the *Mode* field is set to 1, the *Scenes*  
 4273 cluster server SHALL copy all its available scenes with group identifier equal to the *Group Identifier From* field under  
 4274 the group identifier specified in the *Group Identifier To* field, leaving the scene identifiers the same. In this case, the  
 4275 *Scene Identifier From* and *Scene Identifier To* fields are ignored. If a scene already exists under the same group/scene  
 4276 identifier pair, it SHALL be overwritten.

4277 If the *Copy Scene* command was received via the unicast data service, the *Scenes* cluster server SHALL then generate  
 4278 and transmit a *Copy Scene Response* command back to the originator. Otherwise, no response SHALL be given.

4279 **3.7.2.5 Commands Generated**

4280 The generated command IDs for the Scenes cluster are listed in Table 3-43.

4281 **Table 3-43. Generated Command IDs for the Scenes Cluster**

Command Identifier Field Value	Description	M/O
0x00	Add Scene Response	M
0x01	View Scene Response	M
0x02	Remove Scene Response	M
0x03	Remove All Scenes Response	M
0x04	Store Scene Response	M
0x06	Get Scene Membership Response	M
0x40	Enhanced Add Scene Response	O
0x41	Enhanced View Scene Response	O
0x42	Copy Scene Response	O

4282 **3.7.2.5.1 Add Scene Response Command**

4283 **3.7.2.5.1.1 Payload Format**

4284 The Add Scene Response command payload SHALL be formatted as illustrated in Figure 3-28.

4285

**Figure 3-28. Format of the Add Scene Response Command Payload**

<b>Octets</b>	1	2	1
<b>Data Type</b>	enum8	uint16	uint8
<b>Field Name</b>	Status	Group ID	Scene ID

4286 **3.7.2.5.1.2 When Generated**

4287 This command is generated in response to a received Add Scene command 3.7.2.4.2. The Status field is set to  
 4288 SUCCESS, INSUFFICIENT\_SPACE or INVALID\_FIELD (the group is not present in the Group Table) as  
 4289 appropriate. The Group ID and Scene ID fields are set to the corresponding fields of the received Add Scene command.

4290 **3.7.2.5.2 View Scene Response Command**4291 **3.7.2.5.2.1 Payload Format**

4292 The View Scene Response command payload SHALL be formatted as illustrated in Figure 3-29.

4293 **Figure 3-29. Format of the View Scene Response Command Payload**

<b>Octets</b>	1	2	1	0 / 2	0 / Variable	0 / Variable
<b>Data Type</b>	enum8	uint16	uint8	uint16	string	Variable (multiple types)
<b>Field Name</b>	Status	Group ID	Scene ID	Transition time	Scene Name	Extension field sets, one per cluster

4294 The format of each extension field set is a 16 bit field carrying the cluster ID, followed by an 8 bit data length field  
 4295 and the set of scene extension fields specified in the relevant cluster. These fields are concatenated together in the  
 4296 order given in the cluster.

4297 Extension field sets =

4298 {{clusterId 1, length 1, {extension field set 1}}, {clusterId 2, length 2, {extension field set 2}},}.

4299 **3.7.2.5.2.2 When Generated**

4300 This command is generated in response to a received View Scene command 3.7.2.4.3.

4301 The entry in the Scene Table with Scene ID and Group ID given in the received View Scene command is located (if  
 4302 possible). The Status field is set to SUCCESS, NOT\_FOUND (the scene is not present in the Scene Table) or  
 4303 INVALID\_FIELD (the group is not present in the Group Table) as appropriate. The Group ID and Scene ID fields are  
 4304 set to the corresponding fields in the received View Scene command.

4305 If the status is SUCCESS, the Transition time, Scene Name and Extension field fields are copied from the  
 4306 corresponding fields in the table entry, otherwise they are omitted.

4307 **3.7.2.5.3 Remove Scene Response Command**4308 **3.7.2.5.3.1 Payload Format**

4309 The Remove Scene Response command payload SHALL be formatted as illustrated in Figure 3-30.



4310

**Figure 3-30. Format of Remove Scene Response Command Payload**

<b>Octets</b>	1	2	1
<b>Data Type</b>	enum8	uint16	uint8
<b>Field Name</b>	Status	Group ID	Scene ID

4311 **3.7.2.5.3.2 When Generated**

4312 This command is generated in response to a received Remove Scene command 3.7.2.4.4. The Status field is set to  
 4313 SUCCESS, NOT\_FOUND (the scene is not present in the Scene Table) or INVALID\_FIELD (the group is not present  
 4314 in the Group Table) as appropriate. The Group ID and Scene ID fields are set to the corresponding fields of the received  
 4315 Remove Scene command.

4316 **3.7.2.5.3.3 Remove All Scenes Response Command**

4317 **3.7.2.5.3.4 Payload Format**

4318 The Remove All Scenes Response command payload SHALL be formatted as illustrated in Figure 3-31.

4319 **Figure 3-31. Format of the Remove All Scenes Response Command Payload**

<b>Octets</b>	1	2
<b>Data Type</b>	enum8	uint16
<b>Field Name</b>	Status	Group ID

4320 **3.7.2.5.3.5 When Generated**

4321 This command is generated in response to a received Remove All Scenes command, see 3.7.2.4.5. The Status field is  
 4322 set to SUCCESS or INVALID\_FIELD (the group is not present in the Group Table) as appropriate. The Group ID  
 4323 field is set to the corresponding field of the received Remove All Scenes command.

4324 **3.7.2.5.3.6 Store Scene Response Command**

4325 **3.7.2.5.3.7 Payload Format**

4326 The Store Scene Response command payload SHALL be formatted as illustrated in Figure 3-32.

4327 **Figure 3-32. Format of the Store Scene Response Command Payload**

<b>Octets</b>	1	2	1
<b>Data Type</b>	enum8	uint16	uint8
<b>Field Name</b>	Status	Group ID	Scene ID

4328 **3.7.2.5.3.8 When Generated**

4329 This command is generated in response to a received Store Scene command 3.7.2.4.6. The Status field is set to  
 4330 SUCCESS, INSUFFICIENT\_SPACE or INVALID\_FIELD (the group is not present in the Group Table) as  
 4331 appropriate. The Group ID and Scene ID fields are set to the corresponding fields of the received Store Scene  
 4332 command.

4333 **3.7.2.5.3.9 Get Scene Membership Response Command**4334 **3.7.2.5.3.10 Payload Format**

4335 The Get Scene Membership Response command payload SHALL be formatted as illustrated in Figure 3-33.

4336 **Figure 3-33. Format of the Get Scene Membership Response Command Payload**

Octets	1	1	2	0/1	Variable
Data Type	enum8	uint8	uint16	uint8	uint8 x N
Field Name	Status	Capacity	Group ID	Scene count	Scene list

4337 The fields of the get scene membership response command have the following semantics:

4338 The Capacity field SHALL contain the remaining capacity of the scene table of the device (for all groups). The  
4339 following values apply:

4340	0	No further scenes MAY be added.
4341	0 < Capacity < 0xfe	Capacity holds the number of scenes that MAY be added
4342	0xfe	At least 1 further scene MAY be added (exact number is unknown)
4343	0xff	It is unknown if any further scenes MAY be added

4344 The Status field SHALL contain SUCCESS or INVALID\_FIELD (the group is not present in the Group Table) as  
4345 appropriate.

4346 The Group ID field SHALL be set to the corresponding field of the received Get Scene Membership command.

4347 If the status is not SUCCESS, then the Scene count and Scene list field are omitted, else

4348 The Scene count field SHALL contain the number of scenes contained in the Scene list field.

4349 The Scene list field SHALL contain the identifiers of all the scenes in the scene table with the corresponding Group  
4350 ID. If the total number of scenes associated with this Group ID will cause the maximum payload length of a frame to  
4351 be exceeded, then the Scene list field shall contain only as many scenes as will fit.4352 **3.7.2.5.3.11 When Generated**

4353 This command is generated in response to a received Get Scene Membership command, 3.7.2.4.8.

4354 **3.7.2.5.4 Enhanced Add Scene Response Command**4355 The *Enhanced Add Scene Response* command allows a device to respond to an *Enhanced Add Scene* command.4356 The payload of this command SHALL be formatted in the same way as the *Add Scene Response* command, specified  
4357 in the ZCL scenes cluster.4358 **3.7.2.5.5 Enhanced View Scene Response Command**4359 The *Enhanced View Scene Response* command allows a device to respond to an *Enhanced View Scene* command using  
4360 a finer scene transition time that was available in the ZCL.4361 The payload of this command SHALL be formatted in the same way as the *View Scene Response* command, with the  
4362 following difference:4363 The *Transition Time* field SHALL be measured in tenths of a second rather than in seconds.4364 **3.7.2.5.5.1 When Generated**

4365 The *Enhanced View Scene Response* command is generated in response to a received *Enhanced View Scene* command.  
 4366 The entry in the scene table with scene identifier and group identifier given in the received *Enhanced View Scene*  
 4367 command is located (if possible). The *Status* field is set to SUCCESS, NOT\_FOUND (the scene is not present in the  
 4368 scene table) or INVALID\_FIELD (the group is not present in the group table) as appropriate. The group identifier and  
 4369 scene identifier fields are set to the corresponding fields in the received *Enhanced View Scene* command.

4370 If the status is SUCCESS, the *Transition Time*, *Scene Name* and *Extension Field* fields are copied from the  
 4371 corresponding fields in the table entry, otherwise they are omitted.

4372 The *Transition Time* (measured in tenths of a second) SHALL be calculated from the standard transition time field of  
 4373 the scene table entry (measured in seconds) and the new *TransitionTime100ms* field, as specified in this specification.

### 4374 3.7.2.5.6 Copy Scene Response Command

4375 The *Copy Scene Response* command allows a device to respond to a *Copy Scene* command.

4376 The payload of this command SHALL be formatted as illustrated in Figure 3-34.

4377 **Figure 3-34. Format of the Copy Scene Response Command**

Octets	1	2	1
Data Type	uint8	uint16	uint8
Field Name	Status	Group identifier from	Scene identifier from

#### 4378 3.7.2.5.6.1 3.7.2.5.9.1 Status field

4379 The *status* field is 8-bits in length and SHALL contain the status of the copy scene attempt. This field SHALL be set  
 4380 to one of the non-reserved values listed in Table 3-44.

4381 **Table 3-44. Values of the Status Field of the Copy Scene Response Command**

Status Field Value <sup>14</sup>	Description
SUCCESS	Success
INVALID_FIELD	Invalid scene specified
INSUFFICIENT_SPACE	Insufficient space in the scene table

#### 4382 3.7.2.5.6.2 Group Identifier From Field

4383 The *Group Identifier From* field is 16-bits in length and specifies the identifier of the group from which the scene was  
 4384 copied, as specified in the *Copy Scene* command. Together with the *Scene Identifier From* field, this field uniquely  
 4385 identifies the scene that was copied from the scene table.

#### 4386 3.7.2.5.6.3 Scene Identifier From Field

4387 The *Scene Identifier From* field is 8-bits in length and specifies the identifier of the scene from which the scene was  
 4388 copied, as specified in the *Copy Scene* command. Together with the *Group Identifier From* field, this field uniquely  
 4389 identifies the scene that was copied from the scene table.

<sup>14</sup> See Chapter 2 for an enumerated list of status values.

4390 **3.7.2.5.6.4 When Generated**

4391 The *Copy Scene Response* command is generated in response to a received *Copy Scene* command. If, during the copy,  
4392 there is no more space in the scene table for the entire next scene to be copied, the *Status* field SHALL be set to  
4393 INSUFFICIENT\_SPACE and the scenes already copied SHALL be kept. If the group identifier from and scene  
4394 identifier from fields do not specify a scene that exists in the scene table, the *Status* field SHALL be set to  
4395 INVALID\_FIELD. Otherwise, if the copy was successful, the *Status* field SHALL be set to SUCCESS. The group  
4396 identifier from and scene identifier from fields SHALL be set to the same values as in the corresponding fields of the  
4397 received *Copy Scene* command.

4398 **3.7.3 Client**

4399 The Client cluster has no cluster specific attributes. The client generates the cluster specific commands detailed in  
4400 3.7.2.4, as required by the application. The client receives the cluster specific response commands detailed in 3.7.2.5.

4401 **3.8 On/Off**4402 **3.8.1 Overview**

4403 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4404 etc.

4405 Attributes and commands for switching devices between ‘On’ and ‘Off’ states.

4406 **3.8.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added; CCB 1555
2	ZLO 1.0: StartUpOnOff

4407 **3.8.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OO	Type 1 (client to server)

4408 **3.8.1.3 Cluster Identifiers**

Identifier	PICS Code	Name
0x0006	OO	On/Off

4409 **3.8.2 Server**4410 **3.8.2.1 Dependencies**

4411 On receipt of a *Level Control* cluster command that causes the *OnOff* attribute to be set to 0x00, the *OnTime* attribute  
4412 SHALL be set to 0x0000.

4413 On receipt of a *Level Control* cluster command that causes the *OnOff* attribute to be set to 0x01, if the value of the  
 4414 *OnTime* attribute is equal to 0x0000, the device SHALL set the *OffWaitTime* attribute to 0x0000.

### 4415 3.8.2.2 Attributes

4416 The server supports the attributes shown in Table 3-45.

4417 **Table 3-45. Attributes of the On/Off Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>OnOff</i>	bool	0x00 to 0x01	RPS	0x00	M
0x4000	<i>GlobalSceneControl</i>	bool	0x00 to 0x01	R	0x01	O
0x4001	<i>OnTime</i>	uint16	0x0000 to 0xffff	RW	0x0000	O
0x4002	<i>OffWaitTime</i>	uint16	0x0000 to 0xffff	RW	0x0000	O
0x4003	<i>StartUpOnOff</i> <sup>15</sup>	enum8	0x00 to 0xff	RW	MS	O

#### 4418 3.8.2.2.1 OnOff Attribute

4419 The *OnOff* attribute has the following values: 0 = Off, 1 = On.

#### 4420 3.8.2.2.2 GlobalSceneControl Attribute

4421 In order to support the use case where the user gets back the last setting of the devices (e.g. level settings for lamps),  
 4422 a global scene is introduced which is stored when the devices are turned off and recalled when the devices are turned  
 4423 on. The global scene is defined as the scene that is stored with group identifier 0 and scene identifier 0.

4424 The *GlobalSceneControl* attribute is defined in order to prevent a second *off* command storing the all-devices-off  
 4425 situation as a global scene, and to prevent a second *on* command destroying the current settings by going back to the  
 4426 global scene.

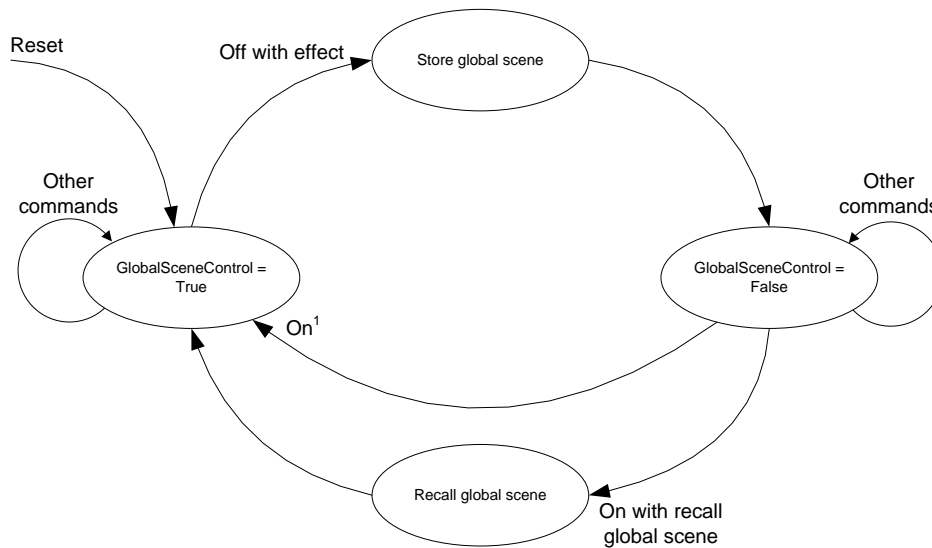
4427 The *GlobalSceneControl* attribute SHALL be set to TRUE after the reception of a command which causes the *OnOff*  
 4428 attribute to be set to TRUE, such as a standard *On* command, a *Move to level (with on/off)* command, a *Recall scene*  
 4429 command or a *On with recall global scene* command (see Section 3.8.2.3.5).

4430 The *GlobalSceneControl* attribute is set to FALSE after reception of a *Off with effect* command.

<sup>15</sup> ZLO 1.0

4431 These concepts are illustrated in Figure 3-35.

4432 **Figure 3-35. State Behavior of Store and Recall Global Scene**



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

4433

4434 **3.8.2.2.3 OnTime Attribute**

4435 The *OnTime* attribute specifies the length of time (in 1/10ths second) that the “on” state SHALL be maintained before  
 4436 automatically transitioning to the “off” state when using the *On with timed off* command. If this attribute is set to  
 4437 0x0000 or 0xffff, the device SHALL remain in its current state.

4438 **3.8.2.2.4 OffWaitTime Attribute**

4439 The *OffWaitTime* attribute specifies the length of time (in 1/10ths second) that the “off” state SHALL be guarded to  
 4440 prevent an on command turning the device back to its “on” state (e.g., when leaving a room, the lights are turned off  
 4441 but an occupancy sensor detects the leaving person and attempts to turn the lights back on). If this attribute is set to  
 4442 0x0000, the device SHALL remain in its current state.

4443 **3.8.2.2.5 StartUpOnOff Attribute**

4444 The *StartUpOnOff* attribute SHALL define the desired startup behavior of a lamp device when it is supplied with  
 4445 power and this state SHALL be reflected in the *OnOff* attribute. The values of the *StartUpOnOff* attribute are listed  
 4446 below.

4447

**Table 3-46. Values of the *StartUpOnOff* Attribute**

Value	Action on power up
0x00	Set the <i>OnOff</i> attribute to 0 (off).
0x01	Set the <i>OnOff</i> attribute to 1 (on).
0x02	If the previous value of the <i>OnOff</i> attribute is equal to 0, set the <i>OnOff</i> attribute to 1. If the previous value of the <i>OnOff</i> attribute is equal to 1, set the <i>OnOff</i> attribute to 0 (toggle).
0x03 to 0xfe	These values are reserved. No action.
0xff	Set the <i>OnOff</i> attribute to its previous value.

4448 **3.8.2.3 Commands Received**

4449 The command IDs for the *On/Off* cluster are listed below.

4450

**Table 3-47. Command IDs for the *On/Off* Cluster**

ID	Description	M/O
0x00	Off	M
0x01	On	M
0x02	Toggle	M
0x40	Off with effect	O
0x41	On with recall global scene	O
0x42	On with timed off	O

4451 **3.8.2.3.1 Off Command**

4452 This command does not have a payload.

4453 **3.8.2.3.1.1 Effect on Receipt**

4454 On receipt of this command, a device SHALL enter its ‘Off’ state. This state is device dependent, but it is  
 4455 recommended that it is used for power off or similar functions. On receipt of the *Off* command, the *OnTime* attribute  
 4456 SHALL be set to 0x0000.

4457 **3.8.2.3.2 On Command**

4458 This command does not have a payload.

4459 **3.8.2.3.2.1 Effect on Receipt**

4460 On receipt of this command, a device SHALL enter its ‘On’ state. This state is device dependent, but it is recommended  
4461 that it is used for power on or similar functions. On receipt of the *On* command, if the value of the *OnTime* attribute  
4462 is equal to 0x0000, the device SHALL set the *OffWaitTime* attribute to 0x0000.

### 4463 3.8.2.3.3 Toggle Command

4464 This command does not have a payload.

#### 4465 3.8.2.3.3.1 Effect on Receipt

4466 On receipt of this command, if a device is in its ‘Off’ state it SHALL enter its ‘On’ state. Otherwise, if it is in its ‘On’  
4467 state it SHALL enter its ‘Off’ state. On receipt of the *Toggle* command, if the value of the *OnOff* attribute is equal to  
4468 0x00 and if the value of the *OnTime* attribute is equal to 0x0000, the device SHALL set the *OffWaitTime* attribute to  
4469 0x0000. If the value of the *OnOff* attribute is equal to 0x01, the *OnTime* attribute SHALL be set to 0x0000.

### 4470 3.8.2.3.4 Off With Effect Command

4471 The *Off With Effect* command allows devices to be turned off using enhanced ways of fading.

4472 The payload of this command SHALL be formatted as illustrated in Figure 3-36.

4473 **Figure 3-36. Format of the Off With Effect Command**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	Effect identifier	Effect variant

#### 4474 3.8.2.3.4.1 Effect Identifier Field

4475 The *Effect Identifier* field is 8-bits in length and specifies the fading effect to use when switching the device off. This  
4476 field SHALL contain one of the non-reserved values listed in Table 3-48.

4477 **Table 3-48. Values of the Effect Identifier Field of the Off With Effect Command**

Effect Identifier Field Value	Description
0x00	Delayed All Off
0x01	Dying Light
0x02 to 0xff	Reserved

#### 4478 3.8.2.3.4.2 Effect Variant Field

4479 The *Effect Variant* field is 8-bits in length and is used to indicate which variant of the effect, indicated in the *Effect*  
4480 *Identifier* field, SHOULD be triggered. If a device does not support the given variant, it SHALL use the default variant.  
4481 This field is dependent on the value of the *Effect Identifier* field and SHALL contain one of the nonreserved values  
4482 listed in Table 3-49.

4483 **Table 3-49. Values of the Effect Variant Field of the Off With Effect Command**

Effect Identifier Field Value	Effect Variant Field Value	Description
0x00	0x00 (default)	Fade to off in 0.8 seconds



Effect Identifier Field Value	Effect Variant Field Value	Description
	0x01	No fade
	0x02	50% dim down in 0.8 seconds then fade to off in 12 seconds
	0x03 to 0xff	Reserved
0x01	0x00 (default)	20% dim up in 0.5s then fade to off in 1 second
	0x01 to 0xff	Reserved
0x02 to 0xff	0x00 to 0xff	Reserved

4484 **3.8.2.3.4.3 Effect on Receipt**

4485 On receipt of the *Off With Effect* command and if the *GlobalSceneControl* attribute is equal to TRUE, the application  
 4486 on the associated endpoint SHALL store its settings in its global scene then set the *GlobalSceneControl* attribute to  
 4487 FALSE. The application SHALL then enter its “off” state, update the *OnOff* attribute accordingly and set the *OnTime*  
 4488 attribute to 0x0000.

4489 In all other cases, the application on the associated endpoint SHALL enter its “off” state and update the *OnOff* attribute  
 4490 accordingly.

4491 **3.8.2.3.5 On With Recall Global Scene Command**

4492 The *On With Recall Global Scene* command allows the recall of the settings when the device was turned off.

4493 The *On With Recall Global Scene* command SHALL have no parameters.

4494 **3.8.2.3.5.1 Effect on Receipt**

4495 On receipt of the *On With Recall Global Scene* command, if the *GlobalSceneControl* attribute is equal to TRUE, the  
 4496 application on the associated endpoint SHALL discard the command.

4497 If the *GlobalSceneControl* attribute is equal to FALSE, the application on the associated endpoint SHALL recall its  
 4498 global scene, entering the appropriate state and updating the *OnOff* attribute accordingly. It SHALL then set the  
 4499 *GlobalSceneControl* attribute to TRUE. In Addition, if the value of the *OnTime* attribute is equal to 0x0000, the device  
 4500 SHALL then set the *OffWaitTime* attribute to 0x0000.

4501 **3.8.2.3.6 On With Timed Off Command**

4502 The *On With Timed Off* command allows devices to be turned on for a specific duration with a guarded off duration  
 4503 so that SHOULD the device be subsequently switched off, further *On With Timed Off* commands, received during this  
 4504 time, are prevented from turning the devices back on. Note that the device can be periodically re-kicked by subsequent  
 4505 *On With Timed Off* commands, e.g., from an on/off sensor.

4506 The payload of this command SHALL be formatted as illustrated in Figure 3-37.

4507 **Figure 3-37. Format of the On With Timed Off Command**

<b>Octets</b>	1	2	2
<b>Data Type</b>	uint8	uint16	uint16
<b>Field Name</b>	On/off Control	On Time	Off Wait Time

4508 **3.8.2.3.6.1 On/Off Control Field**

4509 The *On/Off Control* field is 8-bits in length and contains information on how the device is to be operated. This field  
4510 SHALL be formatted as illustrated in Figure 3-38.

4511 **Figure 3-38. Format of the On/Off Control Field of the On With Timed Off Command**

Bits: 0	1-7
Accept Only When On	Reserved

4512  
4513 The *Accept Only When On* sub-field is 1 bit in length and specifies whether the *On With Timed Off* command is to be  
4514 processed unconditionally or only when the *OnOff* attribute is equal to 0x01. If this sub-field is set to 1, the *On With*  
4515 *Timed Off* command SHALL only be accepted if the *OnOff* attribute is equal to 0x01. If this sub-field is set to 0, the  
4516 *On With Timed Off* command SHALL be processed unconditionally.

#### 4517 **3.8.2.3.6.2 On Time Field**

4518 The *On Time* field is 16 bits in length and specifies the length of time (in 1/10ths second) that the device is to remain  
4519 “on”, i.e., with its *OnOff* attribute equal to 0x01, before automatically turning “off”. This field SHALL be specified  
4520 in the range 0x0000 to 0xffff.

#### 4521 **3.8.2.3.6.3 Off Wait Time Field**

4522 The *Off Wait Time* field is 16 bits in length and specifies the length of time (in 1/10ths second) that the device SHALL  
4523 remain “off”, i.e., with its *OnOff* attribute equal to 0x00, and guarded to prevent an on command turning the device  
4524 back “on”. This field SHALL be specified in the range 0x0000 to 0xffff.

#### 4525 **3.8.2.3.6.4 Effect on Receipt**

4526 On receipt of this command, if the *accept only when on* sub-field of the on/off control field is set to 1 and the value of  
4527 the *OnOff* attribute is equal to 0x00 (off), the command SHALL be discarded.

4528 If the value of the *OffWaitTime* attribute is greater than zero and the value of the *OnOff* attribute is equal to 0x00, then  
4529 the device SHALL set the *OffWaitTime* attribute to the minimum of the *OffWaitTime* attribute and the value specified  
4530 in the off wait time field.

4531 In all other cases, the device SHALL set the *OnTime* attribute to the maximum of the *OnTime* attribute and the value  
4532 specified in the on time field, set the *OffWaitTime* attribute to the value specified in the off wait time field and set the  
4533 *OnOff* attribute to 0x01 (on).

4534 If the values of the *OnTime* and *OffWaitTime* attributes are both less than 0xffff, the device SHALL then update the  
4535 device every 1/10<sup>th</sup> second until both the *OnTime* and *OffWaitTime* attributes are equal to 0x0000, as follows:

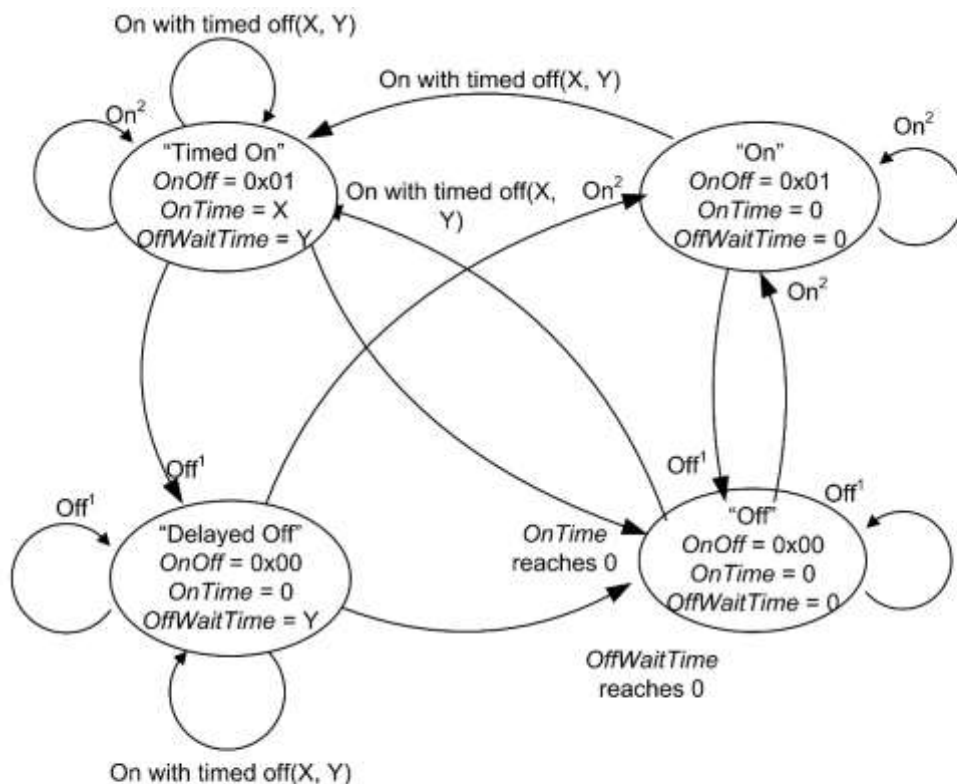
- 4536 • If the value of the *OnOff* attribute is equal to 0x01 (on) and the value of the *OnTime* attribute is greater than  
4537 zero, the device SHALL decrement the value of the *OnTime* attribute. If the value of the *OnTime* attribute  
4538 reaches 0x0000, the device SHALL set the *OffWaitTime* and *OnOff* attributes to 0x0000 and 0x00,  
4539 respectively.
- 4540 • If the value of the *OnOff* attribute is equal to 0x00 (off) and the value of the *OffWaitTime* attribute is greater  
4541 than zero, the device SHALL decrement the value of the *OffWaitTime* attribute. If the value of the *OffWaitTime*  
4542 attribute reaches 0x0000, the device SHALL terminate the update.

### 4543 **3.8.2.4 State Description**

4544 The operation of the on/off cluster with respect to the on, off, and on with timed off commands is illustrated in Figure  
4545 3-39. In this diagram, the values X and Y correspond to the on time and off wait time fields, respectively, of the on  
4546 with timed off command. In the “Timed On” state, the *OnTime* attribute is decremented every 1/10<sup>th</sup> second. Similarly,  
4547 in the “Delayed Off” state, the *OffWaitTime* attribute is decremented every 1/10<sup>th</sup> second.

4548

Figure 3-39. On/Off Cluster Operation State Machine



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

4549

### 4550 3.8.2.5 Commands Generated

4551 The server generates no commands.

### 4552 3.8.2.6 Scene Table Extensions

4553 If the Scenes server cluster (11) is implemented, the following extension field is added to the Scenes table:

4554 *OnOff*

### 4555 3.8.2.7 Attribute Reporting

4556 This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum  
 4557 and maximum reporting interval settings described in Chapter 2, Foundation. The following attribute SHALL be  
 4558 reported:

4559 *OnOff*

## 4560 3.8.3 Client

4561 The client has no cluster specific attributes. The client generates the cluster specific commands received by the server  
 4562 (see 3.8.2.3) , as required by the application. No cluster specific commands are received by the client.

4563 

## 3.9 On/Off Switch Configuration

4564 

### 3.9.1 Overview

4565 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4566 etc.

4567 Attributes and commands for configuring On/Off switching devices.

4568 

#### 3.9.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

4569 

#### 3.9.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OOSC	Type 2 (server to client)

4570 

#### 3.9.1.3 Cluster Identifiers

Identifier	Name
0x0007	On /Off Switch Configuration

4571 

### 3.9.2 Server

4572 

#### 3.9.2.1 Dependencies

4573 Any endpoint that implements this server cluster SHALL also implement the On/Off client cluster.

4574 

#### 3.9.2.2 Attributes

4575 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
4576 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
4577 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
4578 are listed in Table 3-50.4579 **Table 3-50. On/Off Switch Configuration Attribute Sets**

Attribute Set Identifier	Description
0x000	Switch Information
0x001	Switch Settings

4580 **3.9.2.2.1 Switch Information Attribute Set**

4581 The switch information attribute set contains the attributes summarized in Table 3-51.

4582 **Table 3-51. Attributes of the Switch Information Attribute Set**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>SwitchType</i>	enum8	0x00 to 0x01	R	-	M

4583 **3.9.2.2.2 SwitchType Attribute**

4584 The *SwitchType* attribute specifies the basic functionality of the On/Off switching device. This attribute SHALL be  
 4585 set to one of the nonreserved values listed in Table 3-52.

4586 **Table 3-52. Values of the SwitchType Attribute**

Attribute Value	Description	Details
0x00	Toggle	A switch with two physical states. An action by the user (e.g., toggling a rocker switch) moves the switch from state 1 to state 2. The switch then remains in that state until another action from the user returns it to state 1.
0x01	Momentary	A switch with two physical states. An action by the user (e.g., pressing a button) moves the switch from state 1 to state 2. When the user ends his action (e.g., releases the button) the switch returns to state 1.
0x02	Multifunction	A switch that behaves differently depending on user input. Under some conditions it MAY send a toggle or in some other conditions a move command. The behavior of the switch is application-specific but the nature of the switch is clear: it is a multifunction switch.

4587 **3.9.2.2.3 Switch Settings Attribute Set**

4588 The switch settings attribute set contains the attributes summarized in Table 3-53.

4589 **Table 3-53. Attributes of the Switch Settings Attribute Set**

Identifier	Name	Type	Range	Access	Default	M/O
0x0010	<i>SwitchActions</i>	enum8	0x00 to 0x02	RW	0x00	M

4590 **3.9.2.2.3.1 SwitchActions Attribute**

4591 The *SwitchActions* attribute is 8 bits in length and specifies the commands of the On/Off cluster (see 3.8) to be  
 4592 generated when the switch moves between its two states, as detailed in Table 3-54.

4593

Table 3-54. Values of the *SwitchActions* Attribute

Attribute Value	Command Generated When Arriving at State 2 From State 1	Command Generated When Arriving at State 1 From State 2
0x00	On	Off
0x01	Off	On
0x02	Toggle	Toggle

4594 **3.9.2.3 Commands**

4595 No commands are generated or received by the server.

4596 **3.9.3 Client**

4597 The client has no cluster specific attributes. No cluster specific commands are generated or received by the client.

4598 **3.10 Level**4599 **3.10.1 Overview**4600 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4601 etc.4602 This cluster provides an interface for controlling a characteristic of a device that can be set to a level, for example the  
4603 brightness of a light, the degree of closure of a door, or the power output of a heater.4604 NOTE: This cluster specification is a base cluster for generic level control. Also, in this document, is the Level Control  
4605 for Lighting cluster specification, formerly just Level Control. Level Control for Lighting is derived from this cluster  
4606 specification, and has further requirements for the lighting application. Please see section 3.19 for the Level Control  
4607 for Lighting.4608 **3.10.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added
2	added <i>Options</i> attribute, state change table; ZLO 1.0; Base cluster (no change) CCB 2085 1775 2281 2147

4609 **3.10.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	LVL	Type 1 (client to server)

### 4610 3.10.1.3 Cluster Identifiers

4611 Derived cluster specifications are defined elsewhere. This base cluster specification MAY be used for generic level  
 4612 control; however, it is recommended to derive another cluster to better define the application and domain requirements.  
 4613 If one of more derived cluster identifiers and the base identifier exists on a device endpoint, then they SHALL all  
 4614 represent a single instance of the device level control. See Chapter 2 – Instance Model for more information.

Identifier	Hierarchy	Name
0x0008	Base	Level (this cluster specification)
0x0008	Derived	Level Control for Lighting (3.19)
0x001c	Derived	Pulse Width Modulation (3.20)

## 4615 3.10.2 Server

### 4616 3.10.2.1 Dependencies

4617 For many applications, a close relationship between this cluster and the On/Off cluster is needed. This section  
 4618 describes the dependencies that are required when an endpoint that implements this server cluster and also implements  
 4619 the On/Off server cluster.

4620 The *OnOff* attribute of the On/Off cluster and the *CurrentLevel* attribute of the Level Control cluster are intrinsically  
 4621 independent variables, as they are on different clusters. However, when both clusters are implemented on the same  
 4622 endpoint, dependencies MAY be introduced between them. Facilities are provided to introduce dependencies if  
 4623 required.

#### 4624 3.10.2.1.1 Effect of On/Off Commands on the *CurrentLevel* Attribute

4625 The attribute *OnLevel* (see 3.10.2.2.10) determines whether commands of the On/Off cluster have a permanent effect  
 4626 on the *CurrentLevel* attribute or not. If this attribute is defined (i.e., implemented and not 0xff) they do have a  
 4627 permanent effect, otherwise they do not. There is always a temporary effect, due to fading up / down.

4628 The effect on the Level Control cluster on receipt of the various commands of the On/Off cluster are as detailed in  
 4629 Table 3-55. In this table, and throughout this cluster specification, 'level' means the value of the *CurrentLevel* attribute.

4630 **Table 3-55. Actions on Receipt for On/Off Commands, when Associated with Level Control**

Command	Action On Receipt
On	Temporarily store <i>CurrentLevel</i> . Set <i>CurrentLevel</i> to the minimum level allowed for the device. Change <sup>16</sup> <i>CurrentLevel</i> to <i>OnLevel</i> , or to the stored level if <i>OnLevel</i> is not defined, over the time period <i>OnOffTransitionTime</i> .
Off	Temporarily store <i>CurrentLevel</i> . Change <i>CurrentLevel</i> to the minimum level allowed for the device over the time period <i>OnOffTransitionTime</i> . If <i>OnLevel</i> is not defined, set the <i>CurrentLevel</i> to the stored level.
Toggle	If the <i>OnOff</i> attribute has the value Off, proceed as for the On command. Otherwise proceed as for the Off command.

<sup>16</sup> CCB 2085

4631 Intention of the actions described in the table above is that *CurrentLevel*, which was in effect before any of the On,  
4632 Off or Toggle commands were issued, shall be restored, after the transition is completed. If another of these commands  
4633 is received, before the transition is completed, the originally stored *CurrentLevel* shall be preserved and restored<sup>17</sup>.

### 4634 3.10.2.1.2 Effect of Level Control Commands on the *OnOff* Attribute

4635 There are two sets of commands provided in the Level Control cluster. These are identical, except that the first set  
4636 (Move to Level, Move and Step) SHALL NOT affect the *OnOff* attribute, whereas the second set ('with On/Off'  
4637 variants) SHALL.

4638 The first set is used to maintain independence between the *CurrentLevel* and *OnOff* attributes, so changing  
4639 *CurrentLevel* has no effect on the *OnOff* attribute. As examples, this represents the behavior of a volume control with  
4640 a mute button, or a 'turn to set level and press to turn on/off' light dimmer.

4641 The second set is used to link the *CurrentLevel* and *OnOff* attributes. When the level is reduced to its minimum the  
4642 *OnOff* attribute is automatically turned to Off, and when the level is increased above its minimum the *OnOff* attribute  
4643 is automatically turned to On. As an example, this represents the behavior of a light dimmer with no independent  
4644 on/off switch.

### 4645 3.10.2.1.3 GlobalSceneControl and Commands with On/Off

4646 If a *Move to Level (with On/off)*, *Move (with on/Off)* or *Step (with On/Off)* command is received that causes a change  
4647 to the value of the *OnOff* attribute of the On/Off cluster, the value of the *GlobalSceneControl* attribute of the On/Off  
4648 cluster SHALL be updated according to section 3.8.2.2.2.

## 4649 3.10.2.2 Attributes

4650 The attributes of the Level Control server cluster are summarized in Table 3-56.

4651 **Table 3-56. Attributes of the Level Control Server Cluster**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>CurrentLevel</i>	uint8	<i>MinLevel</i> to <i>MaxLevel</i>	RPS	0xff	M
0x0001	<i>RemainingTime</i>	uint16	0x0000 to 0xffff	R	0x0000	O
0x0002	<i>MinLevel</i> <sup>18</sup>	uint8	0x00 to <i>MaxLevel</i>	R	0x00	O
0x0003	<i>MaxLevel</i> <sup>19</sup>	uint8	<i>MinLevel</i> to 0xff	R	0xff	O
0x0004	<i>CurrentFrequency</i> <sup>20</sup>	uint16	<i>MinFrequency</i> to <i>MaxFrequency</i>	RPS	0x0000	O
0x0005	<i>MinFrequency</i> <sup>21</sup>	uint16	0x0000 to <i>MaxFrequency</i>	R	0x0000	O
0x0006	<i>MaxFrequency</i> <sup>22</sup>	uint16	<i>MinFrequency</i> to 0xffff	R	0x0000	O
0x0010	<i>OnOffTransitionTime</i>	uint16	0x0000 to 0xffff	RW	0x0000	O

<sup>17</sup> CCB 1775

<sup>18</sup> NFR Quality of Goods

<sup>19</sup> NFR Quality of Goods

<sup>20</sup> NFR Quality of Goods

<sup>21</sup> NFR Quality of Goods

<sup>22</sup> NFR Quality of Goods



Id	Name	Type	Range	Acc	Default	M/O
0x0011	<i>OnLevel</i>	uint8	<i>MinLevel</i> to <i>MaxLevel</i>	RW	0xff	O
0x0012	<i>OnTransitionTime</i>	uint16	0x0000 to 0xffffe	RW	0xffff	O
0x0013	<i>OffTransitionTime</i>	uint16	0x0000 to 0xffffe	RW	0xffff	O
0x0014	<i>DefaultMoveRate</i>	uint16	0x00 to 0xfe	RW	<i>MS</i>	O
0x000F	<i>Options</i> <sup>23</sup>	map8		RW	0x00	O
0x4000	<i>StartUpCurrentLevel</i> <sup>24</sup>	uint8	0x00 to 0xff	RW	<i>MS</i>	O

### 4652 **3.10.2.2.1** *CurrentLevel* Attribute

4653 The *CurrentLevel* attribute represents the current level of this device. The meaning of 'level' is device dependent.

### 4654 **3.10.2.2.2** *RemainingTime* Attribute

4655 The *RemainingTime* attribute represents the time remaining until the current command is complete - it is specified in  
4656 1/10ths of a second.

### 4657 **3.10.2.2.3** *MinLevel* Attribute

4658 The *MinLevel* attribute indicates the minimum value of *CurrentLevel* that is capable of being assigned.

### 4659 **3.10.2.2.4** *MaxLevel* Attribute

4660 The *MaxLevel* attribute indicates the maximum value of *CurrentLevel* that is capable of being assigned.

### 4661 **3.10.2.2.5** *CurrentFrequency* Attribute

4662 The *CurrentFrequency* attribute represents the frequency that the devices is at *CurrentLevel*. A *CurrentFrequency* of  
4663 0 is unknown.

### 4664 **3.10.2.2.6** *MinFrequency* Attribute

4665 The *MinFrequency* attribute indicates the minimum value of *CurrentFrequency* that is capable of being assigned.  
4666 *MinFrequency* shall be less than or equal to *MaxFrequency*. A value of 0 indicates undefined.

### 4667 **3.10.2.2.7** *MaxFrequency* Attribute

4668 The *MaxFrequency* attribute indicates the maximum value of *CurrentFrequency* that is capable of being assigned.  
4669 *MaxFrequency* shall be greater than or equal to *MinFrequency*. A value of 0 indicates undefined.

<sup>23</sup> CCB 2085

<sup>24</sup> ZLO 1.0

4670 **3.10.2.2.8 Options Attribute<sup>25</sup>**

4671 The *Options* attribute is meant to be changed only during commissioning. The *Options* attribute is a bitmap that  
 4672 determines the default behavior of some cluster commands. Each command that is dependent on the *Options* attribute  
 4673 SHALL first construct a temporary Options bitmap that is in effect during the command processing. The temporary  
 4674 Options bitmap has the same format and meaning as the *Options* attribute, but includes any bits that may be overridden  
 4675 by command fields.

4676 Below is the format and description of the *Options* attribute and temporary Options bitmap and the effect on dependent  
 4677 commands.

4678 **Table 3-57. Options Attribute**

Bit	Name	Values & Summary
0	ExecuteIfOff	0 – Do not execute command if OnOff is 0x00 (FALSE) 1 – Execute command if OnOff is 0x00 (FALSE)
1	<i>Reserved for Derived Clusters</i>	This bit has been defined in these derived clusters for a specific application: Level Control for Lighting

4679

4680 **3.10.2.2.8.1 ExecutelfOff Options Bit**

4681 Command execution SHALL NOT continue beyond the *Options* processing if all of these criteria are true:

- 4682 • The command is one of the ‘without On/Off’ commands: Move, Move to Level, Stop, or Step.
- 4683 • The On/Off cluster exists on the same endpoint as this cluster.
- 4684 • The *OnOff* attribute of the On/Off cluster, on this endpoint, is 0x00 (FALSE).
- 4685 • The value of the ExecuteIfOff bit is 0.

4686

4687 **3.10.2.2.9 OnOffTransitionTime Attribute**

4688 The *OnOffTransitionTime* attribute represents the time taken to move to or from the target level when On of Off  
 4689 commands are received by an On/Off cluster on the same endpoint. It is specified in 1/10ths of a second.

4690 The actual time taken SHOULD be as close to *OnOffTransitionTime* as the device is able. N.B. If the device is not  
 4691 able to move at a variable rate, the *OnOffTransitionTime* attribute SHOULD NOT be implemented.

4692 **3.10.2.2.10 OnLevel Attribute**

4693 The *OnLevel* attribute determines the value that the *CurrentLevel* attribute is set to when the *OnOff* attribute of an  
 4694 On/Off cluster on the same endpoint is set to On, as a result of processing an On/Off cluster command<sup>26</sup>. If the *OnLevel*  
 4695 attribute is not implemented, or is set to 0xff, it has no effect. For more details see 3.10.2.1.1.

---

<sup>25</sup> CCB 2085

<sup>26</sup> CCB 2281 “as a result of processing an On/Off cluster command”

4696 **3.10.2.2.11 OnTransitionTime Attribute**

4697 The *OnTransitionTime* attribute represents the time taken to move the current level from the minimum level to the  
 4698 maximum level when an On command is received by an On/Off cluster on the same endpoint. It is specified in 10ths  
 4699 of a second. If this command is not implemented, or contains a value of 0xffff, the *On/OffTransitionTime* will be used  
 4700 instead.

4701 **3.10.2.2.12 OffTransitionTime Attribute**

4702 The *OffTransitionTime* attribute represents the time taken to move the current level from the maximum level to the  
 4703 minimum level when an Off command is received by an On/Off cluster on the same endpoint. It is specified in 10ths  
 4704 of a second. If this command is not implemented, or contains a value of 0xffff, the *On/OffTransitionTime* will be used  
 4705 instead.

4706 **3.10.2.2.13 DefaultMoveRate Attribute**

4707 The *DefaultMoveRate* attribute determines the movement rate, in units per second, when a Move command is received  
 4708 with a Rate parameter of 0xFF.

4709 **3.10.2.2.14 StartUpCurrentLevel Attribute**

4710 The *StartUpCurrentLevel* attribute SHALL define the desired startup level for a device when it is supplied with power  
 4711 and this level SHALL be reflected in the *CurrentLevel* attribute. The values of the *StartUpCurrentLevel* attribute are  
 4712 listed below:

4713 **Table 3-58. Values of the *StartUpCurrentLevel* attribute**

Value	Action on power up
0x00	Set the <i>CurrentLevel</i> attribute to the minimum value permitted on the device
0xff	Set the <i>CurrentLevel</i> attribute to its previous value
<i>other values</i>	Set the <i>CurrentLevel</i> attribute to this value

4714

4715 **3.10.2.3 Commands Received**

4716 The command IDs for the Level Control cluster are listed below.

4717 **Table 3-59. Command IDs for the Level Control Cluster**

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M

ID	Description	M/O
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency <sup>27</sup>	M: <i>CurrentFrequency</i> attribute supported

### 4718 3.10.2.3.1 Move to Level Command

#### 4719 3.10.2.3.1.1 Payload Format

4720 The Move to Level command payload SHALL be formatted as illustrated in Figure 3-40.

4721 **Figure 3-40. Format of the Move to Level Command Payload**

Octets	1	2	0/1	0/1
Data Type	uint8	uint16	map8	map8
Field Name	Level	Transition time	OptionsMask <sup>28</sup>	OptionsOverride <sup>29</sup>

#### 4722 3.10.2.3.1.2 Effect on Receipt

4723 The OptionsMask & OptionsOverride fields SHALL both be present or both omitted in the command. A temporary  
4724 Options bitmap SHALL be created from the *Options* attribute, using the OptionsMask & OptionsOverride fields, if  
4725 present. Each bit of the temporary Options bitmap SHALL be determined as follows:

4726 Each bit in the *Options* attribute SHALL determine the corresponding bit in the temporary Options bitmap, unless the  
4727 OptionsMask field is present and has the corresponding bit set to 1, in which case the corresponding bit in the  
4728 OptionsOverride field SHALL determine the corresponding bit in the temporary Options bitmap.

4729 The resulting temporary Options bitmap SHALL then be processed as defined in section 3.10.2.2.3.

4730 On receipt of this command, a device SHALL move from its current level to the value given in the Level field. The  
4731 meaning of ‘level’ is device dependent – e.g., for a light it MAY mean brightness level.

4732 The movement SHALL be as continuous as technically practical, i.e., not a step function, and the time taken to move  
4733 to the new level SHALL be equal to the value of the Transition time field, in tenths of a second, or as close to this as  
4734 the device is able.

4735 If the Transition time field takes the value 0xffff then the time taken to move to the new level SHALL instead be  
4736 determined by the *OnOffTransitionTime* attribute. If *OnOffTransitionTime*, which is an optional attribute, is not  
4737 present, the device SHALL move to its new level as fast as it is able.

4738 If the device is not able to move at a variable rate, the Transition time field MAY be disregarded.

### 4739 3.10.2.3.2 Move Command

#### 4740 3.10.2.3.2.1 Payload Format

<sup>27</sup> NFR Quality of Goods

<sup>28</sup> CCB 2085

<sup>29</sup> CCB 2085

4741 The Move command payload SHALL be formatted as illustrated in Figure 3-41.

4742 **Figure 3-41. Format of the Move Command Payload**

<b>Octets</b>	1	1	0/1	0/1
<b>Data Type</b>	enum8	uint8	map8	map8
<b>Field Name</b>	Move mode	Rate	OptionsMask <sup>30</sup>	OptionsOverride <sup>31</sup>

4743 **3.10.2.3.2.2 Move Mode Field**

4744 The Move mode field SHALL be one of the non-reserved values in Table 3-60.

4745 **Table 3-60. Values of the Move Mode Field**

Fade Mode Value	Description
0x00	Up
0x01	Down

4746 **3.10.2.3.2.3 Rate Field**

4747  
 4748 The Rate field specifies the rate of movement in units per second. The actual rate of movement SHOULD be as close  
 4749 to this rate as the device is able. If the Rate field is 0xFF, then the value in *DefaultMoveRate* attribute SHALL be used.  
 4750 If the Rate field is 0xFF and the *DefaultMoveRate* attribute is not supported, then the device SHOULD move as fast  
 4751 as it is able. If the device is not able to move at a variable rate, this field MAY be disregarded.

4752 **3.10.2.3.2.4 Effect on Receipt**

4753 On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in  
 4754 section 3.10.2.3.1.2.

4755 On receipt of this command, a device SHALL move from its current level in an up or down direction in a continuous  
 4756 fashion, as detailed in Table 3-61.

4757 **Table 3-61. Actions on Receipt for Move Command**

Fade Mode	Action on Receipt
Up	Increase the device’s level at the rate given in the Rate field. If the level reaches the maximum allowed for the device, stop.
Down	Decrease the device’s level at the rate given in the Rate field. If the level reaches the minimum allowed for the device, stop.

4758 **3.10.2.3.3 Step Command**

4759 **3.10.2.3.3.1 Payload Format**

<sup>30</sup> CCB 2085

<sup>31</sup> CCB 2085

4760 The Step command payload SHALL be formatted as illustrated in Figure 3-42.

4761 **Figure 3-42. Format of the Step Command Payload**

Octets	1	1	2	0/1	0/1
Data Type	enum8	uint8	uint16	map8	map8
Field Name	Step mode	Step size	Transition time	OptionsMask <sup>32</sup>	OptionsOverride <sup>33</sup>

4762

4763 The Step mode field SHALL be one of the non-reserved values in Table 3-62.

4764 **Table 3-62. Values of the Step Mode Field**

Fade Mode Value	Description
0x00	Up
0x01	Down

4765

4766 The Transition time field specifies the time that SHALL be taken to perform the step, in tenths of a second. A step is  
4767 a change in the *CurrentLevel* of 'Step size' units. The actual time taken SHOULD be as close to this as the device is  
4768 able. If the Transition time field is 0xffff the device SHOULD move as fast as it is able.

4769 If the device is not able to move at a variable rate, the Transition time field MAY be disregarded.

### 4770 **3.10.2.3.3.2 Effect on Receipt**

4771 On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in  
4772 section 3.10.2.3.1.2.

4773 On receipt of this command, a device SHALL move from its current level in an up or down direction as detailed in  
4774 Table 3-63.

4775 **Table 3-63. Actions on Receipt for Step Command**

Fade Mode	Action on Receipt
Up	Increase <i>CurrentLevel</i> by 'Step size' units, or until it reaches the maximum level allowed for the device if this reached in the process. In the latter case, the transition time SHALL be proportionally reduced.
Down	Decrease <i>CurrentLevel</i> by 'Step size' units, or until it reaches the minimum level allowed for the device if this reached in the process. In the latter case, the transition time SHALL be proportionally reduced.

### 4776 **3.10.2.3.4 Stop Command**

#### 4777 **3.10.2.3.4.1 Payload Format**

<sup>32</sup> CCB 2085

<sup>33</sup> CCB 2085

4778 The command payload SHALL be formatted as illustrated below.

4779 **Figure 3-43. Format of the Command Payload**

<b>Octets</b>	0/1	0/1
<b>Data Type</b>	map8	map8
<b>Field Name</b>	OptionsMask <sup>34</sup>	OptionsOverride <sup>35</sup>

4780

4781 **3.10.2.3.4.2 Effect of Receipt**

4782 On receipt of this command, a device SHALL first create and process a temporary Options bitmap as described in  
 4783 section 3.10.2.3.1.2.

4784 Upon receipt of this command, any Move to Level, Move or Step command (and their 'with On/Off' variants) currently  
 4785 in process SHALL be terminated. The value of *CurrentLevel* SHALL be left at its value upon receipt of the Stop  
 4786 command, and *RemainingTime* SHALL be set to zero.

4787 This command has two entries in Table 3-5, one for the Move to Level, Move and Set commands, and one for their  
 4788 'with On/Off' counterparts. This is solely for symmetry, to allow easy choice of one or other set of commands – the  
 4789 Stop commands are identical.

4790 **3.10.2.3.5 Move to Closest Frequency Command**

4791 This command shall be mandatory if the CurrentFrequency attribute is supported.

4792

4793 **3.10.2.3.5.1 Payload Format**

4794 The command payload SHALL be formatted as illustrated below.

4795 **Figure 3-44. Format of the Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	uint16
<b>Field Name</b>	Frequency

4796

4797 **3.10.2.3.5.2 Effect of Receipt**

4798 Upon receipt of this command, the device shall change its current frequency to the requested frequency, or to the  
 4799 closest frequency that it can generate. If the device cannot approximate the frequency, then it shall return a default  
 4800 response with an error code of INVALID\_VALUE. Determining if a requested frequency can be approximated by a  
 4801 supported frequency is a manufacturer-specific decision.

<sup>34</sup> CCB 2085

<sup>35</sup> CCB 2085

**4802 3.10.2.3.6 'With On/Off' Commands**

4803 The Move to Level (with On/Off), Move (with On/Off) and Step (with On/Off) commands have identical payloads to  
4804 the Move to Level, Move and Step commands respectively, except for the OptionsMask and OptionsOverride fields.  
4805 They also have the same effects, except for the following additions.

4806 Before commencing any command that has the effect of setting the *CurrentLevel* above the minimum level allowed  
4807 by the device<sup>36</sup>, the OnOff attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to On.

4808 If any command that has the effect of setting the<sup>37</sup> *CurrentLevel* to the minimum level allowed by the device, the  
4809 OnOff attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to Off.

**4810 3.10.2.4 Commands Generated**

4811 The server generates no commands.

**4812 3.10.3 Client**

4813 The client has no cluster specific attributes. The client generates the cluster specific commands received by the server  
4814 (see 3.10.2.4), as required by the application. No cluster specific commands are received by the client.

**4815 3.11 Alarms****4816 3.11.1 Overview**

4817 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4818 etc.

4819 Attributes and commands for sending alarm notifications and configuring alarm functionality.

4820 Alarm conditions and their respective alarm codes are described in individual clusters, along with an alarm mask field.  
4821 Alarm notifications are reported to subscribed targets using binding.

4822 Where an alarm table is implemented, all alarms, masked or otherwise, are recorded and MAY be retrieved on demand.

4823 Alarms MAY either reset automatically when the conditions that cause are no longer active, or MAY need to be  
4824 explicitly reset.

**4825 3.11.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

**4826 3.11.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ALM	Type 2 (server to client)

<sup>36</sup> CCB 2147 clarify that *CurrentLevel* letting effects On/Off, not increasing or decreasing

<sup>37</sup> CCB 2147 clarify that *CurrentLevel* letting effects On/Off, not increasing or decreasing



4827 **3.11.1.3 Cluster Identifiers**

Identifier	Name
0x0009	Alarms

4828 **3.11.2 Server**

4829 **3.11.2.1 Dependencies**

4830 Any endpoint which implements time stamping SHALL also implement the Time server cluster.

4831 **3.11.2.2 Attributes**

4832 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 4833 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 4834 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 4835 are listed in Table 3-64.

4836 **Table 3-64. Alarms Cluster Attribute Sets**

Attribute Set Identifier	Description
0x000	Alarm Information

4837 **3.11.2.2.1 Alarm Information Attribute Set**

4838 The Alarm Information attribute set contains the attributes summarized in Table 3-65.

4839 **Table 3-65. Attributes of the Alarm Information Attribute Set**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>AlarmCount</i>	uint16	0x00 to maximum defined in profile	R	0x00	O

4840 **3.11.2.2.1.1 AlarmCount Attribute**

4841 The *AlarmCount* attribute is 16 bits in length and specifies the number of entries currently in the alarm table. This  
 4842 attribute SHALL be specified in the range 0x00 to the maximum defined in the profile using this cluster.

4843 If alarm logging is not implemented this attribute SHALL always take the value 0x00.

4844 **3.11.2.3 Alarm Table**

4845 The alarm table is used to store details of alarms generated within the devices. Alarms are requested by clusters which  
 4846 have alarm functionality, e.g., when attributes take on values that are outside ‘safe’ ranges.

4847 The maximum number of entries in the table is device dependent.

4848 When an alarm is generated, a corresponding entry is placed in the table. If the table is full, the earliest entry is replaced  
 4849 by the new entry.

4850 Once an alarm condition has been reported the corresponding entry in the table is removed.

4851 **3.11.2.3.1 Alarm Table Format**

4852 The format of an alarm table entry is illustrated in Table 3-66Format of the Alarm Table.

4853 **Table 3-66. Format of the Alarm Table**

Field	Type	Valid Range	Description
Alarm code	enum8	0x00 to 0xff	Identifying code for the cause of the alarm, as given in the specification of the cluster whose attribute generated this alarm.
Cluster identifier	clusterId	0x0000 to 0xffff	The identifier of the cluster whose attribute generated this alarm.
Time stamp	uint32	0x00000000 to 0xffffffff	The time at which the alarm occurred or 0xffffffff if no time information is available. This time is taken from a Time server cluster, which must be present on the same endpoint.

4854 **3.11.2.4 Commands Received**

4855 The received command IDs for the Alarms cluster are listed in Table 3-67.

4856 **Table 3-67. Received Command IDs for the Alarms Cluster**

Command Identifier Field Value	Description	M/O
0x00	Reset Alarm	M
0x01	Reset all alarms	M
0x02	Get Alarm	O
0x03	Reset alarm log	O

4857 **3.11.2.4.1 Reset Alarm Command**

4858 This command resets a specific alarm. This is needed for some alarms that do not reset automatically. If the alarm  
 4859 condition being reset was in fact still active then a new notification will be generated and, where implemented, a new  
 4860 record added to the alarm log.

4861 **3.11.2.4.1.1 Payload Format**

4862 The Reset Alarm command payload SHALL be formatted as illustrated in Figure 3-45.

4863

**Figure 3-45. Format of the Reset Alarm Command Payload**

Octets	1	2
Data Type	enum8	clusterId
Field Name	Alarm code	Cluster identifier

4864 **3.11.2.4.2 Reset All Alarms Command**

4865 This command resets all alarms. Any alarm conditions that were in fact still active will cause a new notification to be  
 4866 generated and, where implemented, a new record added to the alarm log.

4867 **3.11.2.4.3 Get Alarm Command**

4868 This command causes the alarm with the earliest generated alarm entry in the alarm table to be reported in a get alarm  
 4869 response command 3.11.2.5.2. This command enables the reading of logged alarm conditions from the alarm table.  
 4870 Once an alarm condition has been reported the corresponding entry in the table is removed.  
 4871 This command does not have a payload.

4872 **3.11.2.4.4 Reset Alarm Log Command**

4873 This command causes the alarm table to be cleared, and does not have a payload.

4874 **3.11.2.5 Commands Generated**

4875 The generated command IDs for the Alarms cluster are listed in Table 3-68.

4876 **Table 3-68. Generated Command IDs for the Alarms Cluster**

Command Identifier Field Value	Description	M/O
0x00	Alarm	M
0x01	Get alarm response	O

4877 **3.11.2.5.1 Alarm Command**

4878 The alarm command signals an alarm situation on the sending device.  
 4879 An alarm command is generated when a cluster which has alarm functionality detects an alarm condition, e.g., an  
 4880 attribute has taken on a value that is outside a ‘safe’ range. The details are given by individual cluster specifications.

4881 **3.11.2.5.1.1 Payload Format**

4882 The alarm command payload SHALL be formatted as illustrated in Figure 3-46.

4883

**Figure 3-46. Format of the Alarm Command Payload**

Octets	1	2
Data Type	enum8	clusterId
Field Name	Alarm code	Cluster identifier

4884 **3.11.2.5.2 Get Alarm Response Command**

4885 The get alarm response command returns the results of a request to retrieve information from the alarm log, along  
4886 with a time stamp indicating when the alarm situation was detected.

4887 **3.11.2.5.2.1 Payload Format**

4888 The get alarm response command payload SHALL be formatted as illustrated in Figure 3-47.

4889

**Figure 3-47. Format of the Get Alarm Response Command Payload**

Octets	1	0/1	0/2	0/4
Data Type	enum8	enum8	clusterId	uint32
Field Name	Status	Alarm code	Cluster identifier	Time stamp

4890

4891 If there is at least one alarm record in the alarm table then the status field is set to SUCCESS. The alarm code, cluster  
4892 identifier and time stamp fields SHALL all be present and SHALL take their values from the item in the alarm table  
4893 that they are reporting.

4894 If there are no more alarms logged in the alarm table then the status field is set to NOT\_FOUND and the alarm code,  
4895 cluster identifier and time stamp fields SHALL be omitted.

4896 **3.11.3 Client**

4897 The client has no cluster specific attributes. The client generates the cluster specific commands received by the server  
4898 (see 3.11.2.4), as required by the application. The client receives the cluster specific commands generated by the server  
4899 (see 3.11.2.5).

4900 **3.12 Time**4901 **3.12.1 Overview**

4902 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
4903 etc.

4904 This cluster provides a basic interface to a real-time clock. The clock time MAY be read and also written, in order to  
4905 synchronize the clock (as close as practical) to a time standard. This time standard is the number of seconds since 0  
4906 hrs 0 mins 0 sec on 1<sup>st</sup> January 2000 UTC (Universal Coordinated Time).

4907 The cluster also includes basic functionality for local time zone and daylight saving time.

4908 **3.12.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

4909 **3.12.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Application	T

4910 **3.12.1.3 Cluster Identifiers**

Identifier	Name
0x000a	Time

4911 **3.12.2 Server**

4912 **3.12.2.1 Attributes**

4913 The server supports the attributes shown in Table 3-69.

4914 **Table 3-69. Attributes of the Time Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>Time</i>	UTC	0x00000000 to 0xfffffffffe	RW	0xffffffff	M
0x0001	<i>TimeStatus</i>	map8	0000 xxxx	RW	0b00000000	M
0x0002	<i>TimeZone</i>	int32	-86400 to +86400	RW	0x00000000	O
0x0003	<i>DstStart</i>	uint32	0x00000000 to 0xfffffffffe	RW	0xffffffff	O
0x0004	<i>DstEnd</i>	uint32	0x00000000 to 0xfffffffffe	RW	0xffffffff	O
0x0005	<i>DstShift</i>	int32	-86400 to +86400	RW	0x00000000	O
0x0006	<i>StandardTime</i>	uint32	0x00000000 to 0xfffffffffe	R	0xffffffff	O
0x0007	<i>LocalTime</i>	uint32	0x00000000 to 0xfffffffffe	R	0xffffffff	O
0x0008	<i>LastSetTime</i>	UTC	0x00000000 to 0xffffffff	R	0xffffffff	O
0x0009	<i>ValidUntilTime</i>	UTC	0x00000000 to 0xffffffff	RW	0xffffffff	O

4915 **3.12.2.1.1 Time Attribute**

4916 The *Time* attribute is 32 bits in length and holds the time value of a real time clock. This attribute has data type  
4917 UTCTime, but note that it MAY not actually be synchronized to UTC - see discussion of the *TimeStatus* attribute.

4918 If the Master bit of the *TimeStatus* attribute has a value of 0, writing to this attribute SHALL set the real time clock to  
4919 the written value, otherwise it cannot be written. The value 0xffffffff indicates an invalid time.

4920 **3.12.2.1.2 TimeStatus Attribute**

4921 The *TimeStatus* attribute holds a number of bit fields, as detailed in Table 3-70.

4922 **Table 3-70. Bit Values of the *TimeStatus* Attribute**

Attribute Bit Number	Meaning	Values
0	Master	1 – master clock 0 – not master clock
1	Synchronized	1 – synchronized 0 – not synchronized
2	MasterZoneDst	1 – master for Time Zone and DST 0 – not master for Time Zone and DST
3	Superseding	1 – time synchronization SHOULD be superseded 0 – time synchronization SHOULD not be superseded

4923

4924 The Master and Synchronized bits together provide information on how closely the *Time* attribute conforms to the  
4925 time standard.

4926 The Master bit specifies whether the real time clock corresponding to the *Time* attribute is internally set to the time  
4927 standard. This bit is not writeable – if a value is written to the *TimeStatus* attribute, this bit does not change.

4928 The Synchronized bit specifies whether *Time* has been set over the network to synchronize it (as close as MAY be  
4929 practical) to the time standard (see 3.12.1). This bit must be explicitly written to indicate this – i.e., it is not set  
4930 automatically on writing to the *Time* attribute. If the Master bit is 1, the value of this bit is 0.

4931 If both the Master and Synchronized bits are 0, the real time clock has no defined relationship to the time standard  
4932 (e.g., it MAY record the number of seconds since the device was initialized).

4933 The MasterZoneDst bit specifies whether the *TimeZone*, *DstStart*, *DstEnd* and *DstShift* attributes are set internally to  
4934 correct values for the location of the clock. If not, these attributes need to be set over the network. This bit is not  
4935 writeable – if a value is written to the *TimeStatus* attribute, this bit does not change.

4936 Devices SHALL synchronize to a Time server with the highest rank according to the following rules, listed in order  
4937 of precedence:

4938 • A server with the Superseding bit set SHALL be chosen over a server without the bit set.

4939 • A server with the Master bit SHALL be chosen over a server without the bit set.

- 4940 • The server with the lower short address SHALL be chosen (note that this means a coordinator with the  
4941 Superseding and Master bit set will always be chosen as the network time server).
- 4942 • A Time server with neither the Master nor Synchronized bits set SHOULD not be chosen as the network time  
4943 server.

### 4944 3.12.2.1.3 TimeZone Attribute

4945 The *TimeZone* attribute indicates the local time zone, as a signed offset in seconds from the *Time* attribute value. The  
4946 value 0xffffffff indicates an invalid time zone.

4947 The local Standard Time, i.e., the time adjusted for the time zone, but not adjusted for Daylight Saving Time (DST)  
4948 is given by

4949  $\text{Standard Time} = \text{Time} + \text{TimeZone}$

4950 The range of this attribute is +/- one day. Note that the actual range of physical time zones on the globe is much smaller  
4951 than this, so the manufacturer has the option to impose a smaller range.

4952 If the *MasterZoneDst* bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

### 4953 3.12.2.1.4 DstStart Attribute

4954 The *DstStart* attribute indicates the DST start time in seconds. The value 0xffffffff indicates an invalid DST start time.  
4955 For semantic purposes *DstStart* and *DstEnd* are actually type UTCTime.

4956 The Local Time, i.e., the time adjusted for both the time zone and DST, is given by

4957  $\text{Local Time} = \text{Standard Time} + \text{DstShift}$  (if  $\text{DstStart} \leq \text{Time} \leq \text{DstEnd}$ )

4958  $\text{Local Time} = \text{Standard Time}$  (if  $\text{Time} < \text{DstStart}$  or  $\text{Time} > \text{DstEnd}$ )

4959 Note that the three attributes *DstStart*, *DstEnd* and *DstShift* are optional, but if any one of them is implemented the  
4960 other two must also be implemented.

4961 Note that this attribute SHOULD be set to a new value once every year.

4962 If the *MasterZoneDst* bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

### 4963 3.12.2.1.5 DstEnd Attribute

4964 The *DstEnd* attribute indicates the DST end time in seconds. The value 0xffffffff indicates an invalid DST end time.  
4965 For semantic purposes *DstStart* and *DstEnd* are actually type UTCTime.

4966 Note that this attribute SHOULD be set to a new value once every year, and SHOULD be written synchronously with  
4967 the *DstStart* attribute.

4968 If the *MasterZoneDst* bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

### 4969 3.12.2.1.6 DstShift Attribute

4970 The *DstShift* attribute represents a signed offset in seconds from the standard time, to be applied between the times  
4971 *DstStart* and *DstEnd* to calculate the Local Time (see 3.12.2.1.4). The value 0xffffffff indicates an invalid DST shift.

4972 The range of this attribute is +/- one day. Note that the actual range of DST values employed by countries is much  
4973 smaller than this, so the manufacturer has the option to impose a smaller range.

4974 If the *MasterZoneDst* bit of the *TimeStatus* attribute has a value of 1, this attribute cannot be written.

### 4975 **3.12.2.1.7 StandardTime Attribute**

4976 The local Standard Time is given by the equation in 3.12.2.1.3. Another device on the network MAY calculate this  
4977 time by reading the *Time* and *TimeZone* attributes and adding them together. If implemented however, the optional  
4978 StandardTime attribute indicates this time directly. The value 0xffffffff indicates an invalid Standard Time.

### 4979 **3.12.2.1.8 LocalTime Attribute**

4980 The Local Time is given by the equation in 3.12.2.1.4. Another device on the network MAY calculate this time by  
4981 reading the *Time*, *TimeZone*, *DstStart*, *DstEnd* and *DstShift* attributes and performing the calculation. If implemented  
4982 however, the optional LocalTime attribute indicates this time directly. The value 0xffffffff indicates an invalid Local  
4983 Time.

### 4984 **3.12.2.1.9 LastSetTime Attribute**

4985 The LastSetTime attribute indicates the most recent time that the *Time* attribute was set, either internally or over the  
4986 network (thus it holds a copy of the last value that *Time* was set to). This attribute is set automatically, so is Read  
4987 Only. The value 0xffffffff indicates an invalid LastSetTime.

### 4988 **3.12.2.1.10 ValidUntilTime Attribute**

4989 The ValidUntilTime attribute indicates a time, later than LastSetTime, up to which the Time attribute MAY be trusted.  
4990 ‘Trusted’ means that the difference between the Time attribute and the true UTC time is less than an acceptable error.  
4991 The acceptable error is not defined by this cluster specification, but MAY be defined by the application profile in  
4992 which devices that use this cluster are specified.

4993 **Note:** The value that the ValidUntilTime attribute SHOULD be set to depends both on the acceptable error and the  
4994 drift characteristics of the real time clock in the device that implements this cluster, which must therefore be known  
4995 by the application entity that sets this value.

4996 The value 0xffffffff indicates an invalid ValidUntilTime.

## 4997 **3.12.2.2 Commands Received**

4998 The server receives no commands except those to read and write attributes.

## 4999 **3.12.2.3 Commands Generated**

5000 The server generates no cluster specific commands.

## 5001 **3.12.3 Client**

5002 The client has no cluster specific attributes. No cluster specific commands are generated or received by the client.

# 5003 **3.13 RSSI Location**

---

## 5004 **3.13.1 Overview**

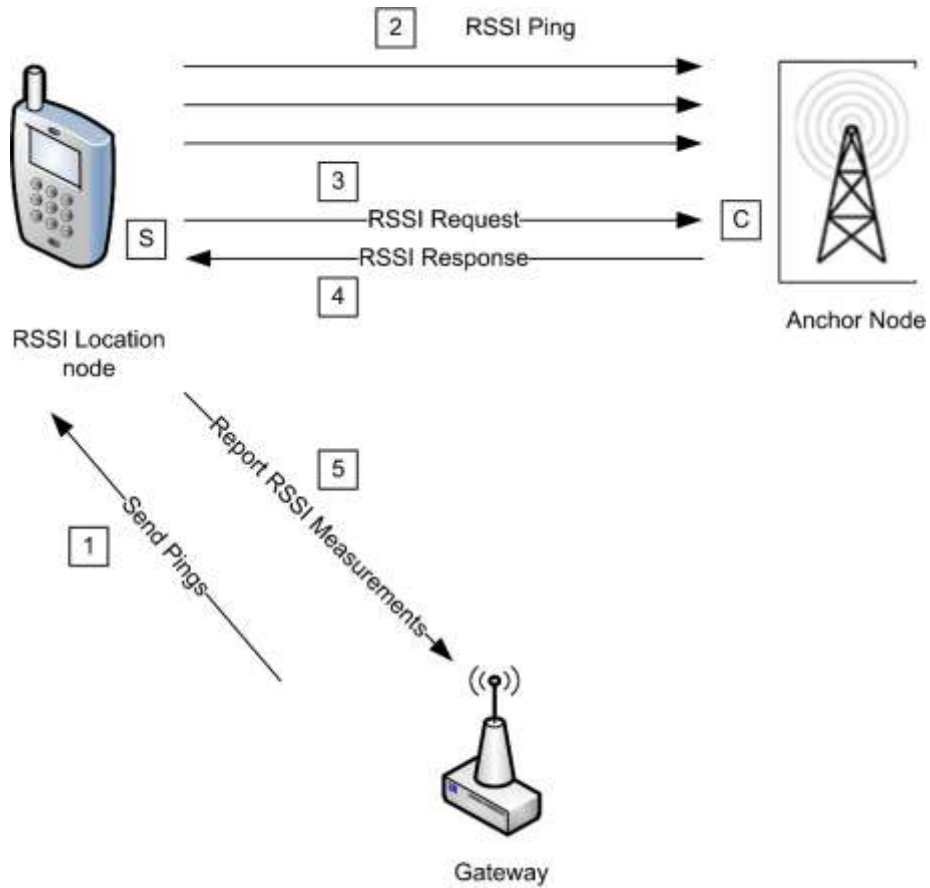
5005 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
5006 etc.

5007 This cluster provides a means for exchanging Received Signal Strength Indication (RSSI) information among one hop  
5008 devices as well as messages to report RSSI data to a centralized device that collects all the RSSI data in the network.  
5009 An example of the usage of RSSI location cluster is shown in Figure 3-48.



5010

Figure 3-48. Example of Usage of RSSI Location Cluster



5011

5012 **3.13.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

5013 **3.13.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	RSSI

5014 **3.13.1.3 Cluster Identifiers**

Identifier	Name
0x000b	RSSI

5015 **3.13.2 Server**5016 **3.13.2.1 Attributes**

5017 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
5018 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
5019 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
5020 are listed in Table 3-71.

5021 **Table 3-71. Location Attribute Sets**

Attribute Set Identifier	Description
0x000	Location Information
0x001	Location Settings

5022 **3.13.2.1.1 Location Information Attribute Set**

5023 The Location Information attribute set contains the attributes summarized in Table 3-72.

5024 **Table 3-72. Attributes of the Location Information Attribute Set**

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	<i>LocationType</i>	data8	0000xxxx	RW	-	M
0x0001	<i>LocationMethod</i>	enum8	0x00 to 0xff	RW	-	M
0x0002	<i>LocationAge</i>	uint16	0x0000 to 0xffff	R	-	O
0x0003	<i>QualityMeasure</i>	uint8	0x00 to 0x64	R	-	O
0x0004	<i>NumberOfDevices</i>	uint8	0x00 to 0xff	R	-	O

5025 **3.13.2.1.1.1 LocationType Attribute**

5026 The *LocationType* attribute is 8 bits long and is divided into bit fields. The meanings of the individual bit fields are  
5027 detailed in Table 3-73.

5028 **Table 3-73. Bit Values of the LocationType Attribute**

Bit Field (Bit Numbers)	Meaning	Values
0	Absolute	1 – Absolute location 0 – Measured location
1	2-D	1 – Two dimensional 0 – Three dimensional
2-3	Coordinate System	0 – Rectangular (installation-specific origin and orientation)

5029 The Absolute bit field indicates whether the location is a known absolute location or is calculated.

5030 The 2-D bit field indicates whether the location information is two- or three-dimensional. If the location information  
 5031 is two-dimensional, Coordinate 3 is unknown and SHALL be set to 0x8000.

5032 The Coordinate System bit field indicates the geometry of the system used to express the location coordinates. If the  
 5033 field is set to zero, the location coordinates are expressed using the rectangular coordinate system. All other values  
 5034 are reserved.

5035 **3.13.2.1.1.2 LocationMethod Attribute**

5036 The *LocationMethod* attribute SHALL be set to one of the non-reserved values in Table 3-74.

5037 **Table 3-74. Values of the *LocationMethod* Attribute**

Value	Method	Description
0x00	Lateration	A method based on RSSI measurements from three or more sources.
0x01	Signposting	The location reported is the location of the neighboring device with the strongest received signal.
0x02	RF fingerprinting	RSSI signatures are collected into a database at commissioning time. The location reported is the location taken from the RSSI signature database that most closely matches the device's own RSSI signature.
0x03	Out of band	The location is obtained by accessing an out-of-band device (that is, the device providing the location is not part of the network).
0x04	Centralized	The location is performed in a centralized way (e.g., by the GW) by a device on the network. Different from the above because the device performing the localization is part of the network.
0x40 to 0xff	-	Reserved for manufacturer specific location methods.

5038 **3.13.2.1.1.3 LocationAge Attribute**

5039 The *LocationAge* attribute indicates the amount of time, measured in seconds, that has transpired since the location  
 5040 information was last calculated. This attribute is not valid if the Absolute bit of the *LocationType* attribute is set to  
 5041 one.

5042 **3.13.2.1.1.4 QualityMeasure Attribute**

5043 The *QualityMeasure* attribute is a measure of confidence in the corresponding location information. The higher the  
 5044 value, the more confident the transmitting device is in the location information. A value of 0x64 indicates complete  
 5045 (100%) confidence and a value of 0x00 indicates zero confidence. (Note: no fixed confidence metric is mandated –  
 5046 the metric MAY be application and manufacturer dependent.)

5047 This field is not valid if the Absolute bit of the *LocationType* attribute is set to one.

5048 **3.13.2.1.1.5 NumberOfDevices Attribute**

5049 The *NumberOfDevices* attribute is the number of devices whose location data were used to calculate the last location  
 5050 value. This attribute is related to the *QualityMeasure* attribute.

5051 **3.13.2.1.2 Location Settings Attribute Set**

5052 The Location Settings attribute set contains the attributes summarized in Table 3-75.

5053

**Table 3-75. Attributes of the Location Settings Attribute Set**

Identifier	Name	Type	Range	Access	Def	M/O
0x0010	<i>Coordinate1</i>	int16	0x8000 to 0x7fff	RW	-	M
0x0011	<i>Coordinate2</i>	int16	0x8000 to 0x7fff	RW	-	M
0x0012	<i>Coordinate3</i>	int16	0x8000 to 0x7fff	RW	-	O
0x0013	<i>Power</i>	int16	0x8000 to 0x7fff	RW	-	M
0x0014	<i>PathLossExponent</i>	uint16	0x0000 to 0xffff	RW	-	M
0x0015	<i>ReportingPeriod</i>	uint16	0x0000 to 0xffff	RW	-	O
0x0016	<i>CalculationPeriod</i>	uint16	0x0000 to 0xffff	RW	-	O
0x0017	<i>NumberRSSIMeasurements</i>	uint8	0x01 to 0xff	RW	-	M

5054 **3.13.2.1.2.1 Coordinate 1,2,3 Attributes**

5055 The *Coordinate1*, *Coordinate2* and *Coordinate3* attributes are signed 16-bit integers, and represent orthogonal linear  
5056 coordinates x, y, z in meters as follows.

5057 
$$x = \text{Coordinate1} / 10, y = \text{Coordinate2} / 10, z = \text{Coordinate3} / 10$$

5058 The range of x is -3276.7 to 3276.7 meters, corresponding to *Coordinate1* between 0x8001 and 0x7fff. The same  
5059 range applies to y and z. A value of 0x8000 for any of the coordinates indicates that the coordinate is unknown.

5060 **3.13.2.1.2.2 Power Attribute**

5061 The *Power* attribute specifies the value of the average power  $P_0$ , measured in dBm, received at a reference distance of  
5062 one meter from the transmitter.

5063 
$$P_0 = \text{Power} / 100$$

5064 A value of 0x8000 indicates that *Power* is unknown.

5065 **3.13.2.1.2.3 PathLossExponent Attribute**

5066 The *PathLossExponent* attribute specifies the value of the Path Loss Exponent n, an exponent that describes the rate  
5067 at which the signal power decays with increasing distance from the transmitter.

5068 
$$n = \text{PathLossExponent} / 100$$

5069 A value of 0xffff indicates that *PathLossExponent* is unknown.

5070 The signal strength in dBm at a distance d meters from the transmitter is given by

5071 
$$P = P_0 - 10n \times \log_{10}(d)$$

5072 where

5073 P is the power in dBm at the receiving device.

5074  $P_0$  is the average power in dBm received at a reference distance of 1 meter from the transmitter.

5075 n is the path loss exponent.

5076 d is the distance in meters between the transmitting device and the receiving device.

5077 **3.13.2.1.2.4 ReportingPeriod Attribute**

5078 The *ReportingPeriod* attribute specifies the time in seconds between successive reports of the device's location by  
 5079 means of the Location Data Notification command. The minimum value this attribute can take is specified by the  
 5080 profile in use. If *ReportingPeriod* is zero, the device does not automatically report its location. Note that location  
 5081 information can always be polled at any time.

5082 **3.13.2.1.2.5 CalculationPeriod Attribute**

5083 The *CalculationPeriod* attribute specifies the time in milliseconds between successive calculations of the device's  
 5084 location. If *CalculationPeriod* is less than the physically possible minimum period that the calculation can be  
 5085 performed, the calculation will be repeated as frequently as possible. In case of centralized location (*LocationMethod*  
 5086 attribute equal to Centralized) the *CalculationPeriod* attribute specifies the period between successive RSSI ping  
 5087 commands.

5088 **3.13.2.1.2.6 NumberRSSIMeasurements Attribute**

5089 The *NumberRSSIMeasurements* attribute specifies the number of RSSI measurements to be used to generate one  
 5090 location estimate. The measurements are averaged to improve accuracy. *NumberRSSIMeasurements* must be greater  
 5091 than or equal to 1. In the case of centralized location (*LocationMethod* attribute equal to Centralized) the  
 5092 *NumberRSSIMeasurements* attribute specifies the number of successive RSSI Ping commands to be sent by the server  
 5093 side of location cluster.

5094 **3.13.2.2 Commands Received**

5095 The received command IDs for the Location cluster are listed in Table 3-76.

5096 **Table 3-76. Received Command IDs for the Location Cluster**

Command Identifier Field Value	Description	M/O
0x00	Set Absolute Location	M
0x01	Set Device Configuration	M
0x02	Get Device Configuration	M
0x03	Get Location Data	M
0x04	RSSI Response	O
0x05	Send Pings	O
0x06	Anchor Node Announce	O

5097 **3.13.2.2.1 Set Absolute Location Command**

5098 This command is used to set a device's absolute (known, not calculated) location and the channel parameters  
 5099 corresponding to that location.

5100 **3.13.2.2.1.1 Payload Format**

5101 The Set Absolute Location command payload SHALL be formatted as illustrated in Figure 3-49.

5102

**Figure 3-49. Format of the Set Absolute Location Command Payload**

<b>Octets</b>	2	2	2	2	2
<b>Data Type</b>	int16	int16	int16	int16	uint16
<b>Field Name</b>	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent

5103 The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and  
5104 ranges see the descriptions of the individual attributes.

5105 The three coordinate fields SHALL contain the absolute location (known, not calculated) of the destination device. If  
5106 any coordinate field(s) is not known, the value(s) SHALL be set to 0x8000.

#### 5107 **3.13.2.2.1.2 Effect on Receipt**

5108 On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as)  
5109 the payload fields.

### 5110 **3.13.2.2.2 Set Device Configuration Command**

5111 This command is used to set a device's location parameters, which will be used for calculating and reporting measured  
5112 location. This command is invalid unless the Absolute bit of the *LocationType* attribute has a value of 0.

#### 5113 **3.13.2.2.2.1 Payload Format**

5114 The Set Device Configuration command payload SHALL be formatted as illustrated in Figure 3-50.

5115

**Figure 3-50. Format of the Set Device Configuration Payload**

<b>Octets</b>	2	2	2	1	2
<b>Data Type</b>	int16	uint16	uint16	uint8	uint16
<b>Field Name</b>	Power	Path Loss Exponent	Calculation Period	Number RSSI Measurements	Reporting Period

5116

5117 The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and  
5118 ranges see the descriptions of the individual attributes.

#### 5119 **3.13.2.2.2.2 Effect on Receipt**

5120 On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as)  
5121 the payload fields.

### 5122 **3.13.2.2.3 Get Device Configuration Command**

5123 This command is used to request the location parameters of a device. The location parameters are used for calculating  
5124 and reporting measured location.

#### 5125 **3.13.2.2.3.1 Payload Format**

5126 The Get Device Configuration command payload SHALL be formatted as illustrated in Figure 3-51.

5127 **Figure 3-51. Format of the Get Device Configuration Payload**

<b>Octets</b>	8
<b>Data Type</b>	EUI64
<b>Field Name</b>	Target Address

5128  
 5129 The Target Address field contains the 64-bit IEEE address of the device for which the location parameters are being  
 5130 requested. This field MAY contain the address of the sending device, the address of the receiving device or the address  
 5131 of a third device.

5132 **Note:** one reason a device MAY request its own configuration is that there MAY be a designated device which holds  
 5133 the configurations of other devices for distribution at commissioning time. It is also possible that the device MAY  
 5134 lose its configuration settings for some other reason (loss of power, reset). In the case of a third device, that device  
 5135 MAY sleep a lot and not be easily accessible.

5136 **3.13.2.2.3.2 Effect on Receipt**

5137 On receipt of this command, the device SHALL generate a Device Configuration Response command (3.13.2.3.1).

5138 **3.13.2.2.4 Get Location Data Command**

5139 This command is used to request a device’s location information and channel parameters. It MAY be sent as a unicast,  
 5140 multicast or broadcast frame. When sent as a broadcast frame, care SHOULD be taken to minimize the risk of a  
 5141 broadcast 'storm' - in particular, it is recommended that the broadcast radius is set to 1.

5142 (**Note:** devices MAY or MAY not acquire and store information on other devices' locations such that this information  
 5143 MAY be requested by another device. This is application dependent.)

5144 **3.13.2.2.4.1 Payload Format**

5145 The Get Location Data command payload SHALL be formatted as illustrated in Figure 3-52.

5146 **Figure 3-52. Format of the Get Location Data Payload**

<b>Bits</b>	3	1	1	1	1	1	8	0 / 64
<b>Data Type</b>	map8						uint8	EUI64
<b>Field Name</b>	Reserved	Compact Response	Broadcast Response	Broadcast Indicator	Recalculate	Absolute Only	Number Responses	Target Address

5147  
 5148 The highest 3 bits of the first octet are reserved and SHALL be set to zero.

5149 The Absolute Only field (bit 0 of the first octet) specifies the type of location information being requested. If the  
5150 Absolute Only field is set to one, the device is only requesting absolute location information (a device MAY want to  
5151 gather absolute node locations for use in its own location calculations, and MAY not be interested in neighbors with  
5152 calculated values). Otherwise, if the field is set to zero, the device is requesting all location information (absolute and  
5153 calculated).

5154 The Recalculate field (bit 1 of the first octet) indicates whether the device is requesting that a new location calculation  
5155 be performed. If the field is set to zero, the device is requesting the currently stored location information. Otherwise,  
5156 if the field is set to one, the device is requesting that a new calculation be performed. This field is only valid if the  
5157 Absolute Only field is set to zero.

5158 The Broadcast Indicator field (bit 2 of the first octet) indicates whether the command is being sent as a unicast,  
5159 multicast or broadcast frame. If the field is set to one, the command is sent as a broadcast or multicast, else it is sent  
5160 as a unicast.

5161 The Broadcast Response field (bit 3 of the first octet) indicates whether subsequent responses after the first (where  
5162 the Number Responses field is greater than one) SHALL be unicast or broadcast. Broadcast responses can be used as  
5163 a 'location beacon'.

5164 The Compact Response field (bit 4 of the first octet) indicates whether subsequent responses after the first (where the  
5165 Number Responses field is greater than one) SHALL be sent using the Location Data Notification or the Compact  
5166 Location Data Notification command.

5167 The Number Responses field indicates the number of location responses to be returned. The information to be returned  
5168 is evaluated this number of times, with a period equal to the value of the *ReportingPeriod* attribute, and a separate  
5169 response is sent for each evaluation. This field SHALL have a minimum value of one. Values greater than one are  
5170 typically used for situations where locations are changing.

5171 The Target Address field contains the 64-bit IEEE address of the device for which the location information and channel  
5172 parameters are being requested. If the Broadcast Indicator field is set to zero (i.e., the command is sent as a unicast)  
5173 this field MAY contain the address of the receiving device, the address of the sending device or the address of any  
5174 other device. If the Broadcast Indicator field is set to one (i.e., the command is sent as a broadcast or multicast) the  
5175 target address is implicitly that of the receiving device, so this field SHALL be omitted.

#### 5176 **3.13.2.2.4.2 Effect on Receipt**

5177 On receipt of this command, if the Location Type field is set to zero, only a receiving device(s) that knows its absolute  
5178 location SHALL respond by generating a Location Data Response command. If the Location Type field is set to one,  
5179 all devices receiving this command SHALL respond by generating a Location Data Response command.

5180 If the command is sent as a unicast, information for the device specified in the Target Address field SHALL be  
5181 returned, if the receiving device has or can obtain the information for that device. If the information is not available,  
5182 the Status field of the Location Data Response command SHALL be set to NOT\_FOUND.

5183 If the command is sent as a broadcast or multicast, receiving devices SHALL send back their own information (there  
5184 is no IEEE target address in this case).

5185 If the Number Responses field is greater than one, the subsequent location readings/calculations SHALL be sent using  
5186 the Location Data Notification or the Compact Location Data Notification command, depending on the value of the  
5187 Reduced Response field.

#### 5188 **3.13.2.2.5 RSSI Response Command**

5189 This command is sent by a device in response to an RSSI Request command.

##### 5190 **3.13.2.2.5.1 Payload Format**

5191 The command payload SHALL be formatted as illustrated in Figure 3-53.



5192 **Figure 3-53. Format of the RSSI Response Command Payload**

<b>Octets</b>	8	2	2	2	1	1
<b>Data Type</b>	EUI64	int16	int16	int16	int8	uint8
<b>Field Name</b>	Replying Device	X	Y	Z	RSSI	<i>NumberRSSIMeasurements</i>

5193  
 5194 The fields of the payload have the following meanings:  
 5195 Replying Device: The IEEE address of the neighbor that replies to the RSSI request  
 5196 X, Y, Z: The coordinates of the replying node  
 5197 RSSI: The RSSI registered by the replying node that refers to the radio link, expressed in dBm, between itself and the  
 5198 neighbor that performed the RSSI request  
 5199 *NumberRSSIMeasurements*: How many packets were considered to give the RSSI value (=1 meaning no mean is  
 5200 supported)

5201 **3.13.2.2.5.2 Effect on Receipt**

5202 On receipt of this command, the server side of the location cluster will wait for *CalculationPeriod* time and generate  
 5203 a Report RSSI Measurement command.

5204 **3.13.2.2.6 Send Pings Command**

5205 This command is used to alert a node to start sending multiple packets so that all its one-hop neighbors can calculate  
 5206 the mean RSSI value of the radio link.

5207 **3.13.2.2.6.1 Payload Format**

5208 Send Pings command SHALL be formatted as illustrated in Figure 3-54. The address field contains the IEEE address  
 5209 of the node that have to perform the blasting (the destination node of this command) and the other fields of the payload  
 5210 correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions  
 5211 of the individual attributes.

5212 **Figure 3-54. Format of the Send Pings Command Payload**

<b>Octets</b>	8	1	2
<b>Data Type</b>	EUI64	uint8	uint16
<b>Field Name</b>	Target Address	<i>NumberRSSIMeasurements</i>	<i>CalculationPeriod</i>

5213  
 5214 The Target Address field contains the IEEE address of the intended target node. This is included because there can be  
 5215 cases when the sender does not definitely know the short address of the intended target (see below for effect on  
 5216 receipt). The other fields of the payload correspond directly to the attributes with the same names. For details of their  
 5217 meaning and ranges see the descriptions of the individual attributes.

5218 **3.13.2.2.6.2 Effect on Receipt**

5219 On receipt of this command, the device SHALL update the attributes corresponding to (i.e., with the same names as)  
5220 the payload fields and generate a number of RSSI Ping commands equal to *NumberRSSIMeasurements* waiting for  
5221 *CalculationPeriod* time between successive transmission of pings.

### 5222 3.13.2.2.7 Anchor Node Announce Command

5223 This command is sent by an anchor node when it joins the network, if it is already commissioned with the coordinates,  
5224 to announce itself so that the central device knows the exact position of that device. This message SHOULD be either  
5225 unicast to the central node or broadcast in the case that of unknown destination address.

#### 5226 3.13.2.2.7.1 Payload Format

5227 Into the payload there are both the short and long addresses of the joining node as well as the coordinates of the node  
5228 itself. 0xffff SHOULD be used if coordinates are not known.

5229 The command payload SHALL be formatted as in Figure 3-55.

5230 **Figure 3-55. Format of the Anchor Node Announce Command Payload**

<b>Octets</b>	8	2	2	2
<b>Data Type</b>	EUI64	int16	int16	int16
<b>Field Name</b>	Anchor Node IEEE Address	X	Y	Z

5231  
5232 The Anchor Node Address field contains the IEEE address of the anchor node. The other fields of the payload  
5233 correspond directly to the attributes with the same names. For details of their meaning and ranges see the descriptions  
5234 of the individual attributes. If any coordinate is unknown, it SHOULD be set to 0x8000.

### 5235 3.13.2.3 Commands Generated

5236 **Table 3-77. Generated Command IDs for the RSSI Location Cluster**

<b>Command Identifier Field Value</b>	<b>Description</b>	<b>M/O</b>
0x00	Device configuration response	M
0x01	Location data response	M
0x02	Location data notification	M
0x03	Compact location data notification	M
0x04	RSSI Ping	M
0x05	RSSI Request	O
0x06	Report RSSI Measurements	O

Command Identifier Field Value	Description	M/O
0x07	Request Own Location	O

5237 **3.13.2.3.1 Device Configuration Response Command**

5238 This command is sent by a device in response to a Get Device Configuration command (3.13.2.2.3).

5239 **3.13.2.3.1.1 Payload Format**

5240 The Device Configuration Response command payload SHALL be formatted as illustrated in Figure 3-56. All payload  
 5241 fields are relevant to the device for which the location parameters have been requested.

5242 **Figure 3-56. Format of the Device Configuration Response Payload**

<b>Octets</b>	1	0 / 2	0 / 2	0 / 2	0 / 1	0 / 2
<b>Data Type</b>	enum8	int16	uint16	uint16	uint8	uint16
<b>Field Name</b>	Status	Power	Path Loss Exponent	Calculation Period	Number RSSI Measurements	Reporting Period

5243

5244 The fields of the payload (other than Status) correspond directly to the attributes with the same names. For details of  
 5245 their meaning and ranges see the descriptions of the individual attributes.

5246 The Status field indicates whether the response to the request was successful or not. If the field is set to SUCCESS,  
 5247 the response was successful. If the field is set to NOT\_FOUND, the receiving device was unable to provide the location  
 5248 parameters of the device for which the location parameters were requested. If the field is set to NOT\_FOUND, all  
 5249 other payload fields SHALL NOT be sent.

5250 **3.13.2.3.2 Location Data Response Command**

5251 This command is sent by a device in response to a request for location information and channel parameters.

5252 **3.13.2.3.2.1 Payload Format**

5253 The Location Data Response command payload SHALL be formatted as illustrated in Figure 3-57. All payload fields  
 5254 are relevant to the device for which the location parameters have been requested.

5255

**Figure 3-57. Format of the Location Data Response Payload**

<b>Octets</b>	1	0 / 1	0 / 2	0 / 2	0 / 2	0 / 2	0 / 2	0 / 1	0 / 1	0 / 2
<b>Data Type</b>	enum8	data8	int16	int16	int16	int16	uint16	enum8	uint8	uint16
<b>Field Name</b>	Status	Location Type	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent	Location Method	Quality Measure	Location Age

5256

5257 The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and  
5258 ranges see the descriptions of the individual attributes.

5259 If the Absolute bit of the Location Type field is set to 1, the Location Method, Quality Measure and Location Age  
5260 fields are not applicable and SHALL NOT be sent.

5261 If the 2-D bit of the Location Type field is set to 1, the Coordinate 3 field SHALL NOT be sent.

5262 The Status field indicates whether the response to the request was successful or not. If the field is set to SUCCESS,  
5263 the response was successful. If the field is set to NOT\_FOUND, the receiving device was unable to provide the location  
5264 parameters of the device for which the location parameters were requested. If the field is set to NOT\_FOUND, all  
5265 other payload fields SHALL NOT be sent.

### 5266 3.13.2.3.3 Location Data Notification Command

5267 This command is sent periodically by a device to announce its location information and channel parameters. The  
5268 period is equal to the value of the *ReportingPeriod* attribute.

5269 The location data notification command MAY be sent as a unicast or as a broadcast frame. When sent as a broadcast  
5270 frame, it is recommended that the broadcast radius is set to 1.

#### 5271 3.13.2.3.3.1 Payload Format

5272 The Location Data Notification command payload SHALL be formatted as illustrated in Figure 3-58.

5273

**Figure 3-58. Format of the Location Data Notification Payload**

<b>Octets</b>	1	2	2	0 / 2	2	2	0 / 1	0 / 1	0 / 2
<b>Data Type</b>	data8	int16	int16	int16	int16	uint16	enum8	uint8	uint16
<b>Field Name</b>	Location Type	Coordinate 1	Coordinate 2	Coordinate 3	Power	Path Loss Exponent	Location Method	Quality Measure	Location Age

5274

5275 The fields of the payload correspond directly to the attributes with the same names. For details of their meaning and  
 5276 ranges see the descriptions of the individual attributes.

5277 If the 2-D bit of the Location Type field is set to 1, the Coordinate 3 field SHALL NOT be sent.

5278 If the Absolute bit of the Location Type field is set to 1, the Location Method, Quality Measure and Location Age  
 5279 fields are not applicable and SHALL NOT be sent.

5280 **3.13.2.3.4 Compact Location Data Notification Command**

5281 This command is identical in format and use to the Location Data Notification command, except that the Power, Path  
 5282 Loss Exponent and Location Method fields are not included.

5283 **3.13.2.3.5 RSSI Ping Command**

5284 This command is sent periodically by a device to enable listening devices to measure the received signal strength in  
 5285 the absence of other transmissions from that device. The period is given by the *ReportingPeriod* attribute.

5286 The RSSI Ping command MAY be sent as a unicast or as a broadcast frame. When sent as a broadcast frame, it is  
 5287 recommended that the broadcast radius is set to 1.

5288 **3.13.2.3.5.1 Payload Format**

5289 The RSSI Ping command payload SHALL be formatted as illustrated in Figure 3-59.

5290 **Figure 3-59. Format of the RSSI Ping Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	data8
<b>Field Name</b>	Location Type

5291  
 5292 The Location Type field holds the value of the *LocationType* attribute.

5293 **3.13.2.3.6 RSSI Request Command**

5294 A device uses this command to ask one, more, or all its one-hop neighbors for the (mean) RSSI value they hear from  
 5295 itself.

5296 **3.13.2.3.6.1 Payload Format**

5297 The message is empty and MAY be used in broadcast (typical usage is broadcast with radius equal to one).

5298 **3.13.2.3.6.2 Effect on Receipt**

5299 On receipt of this command, the device SHALL respond by generating an RSSI Response command back to the sender  
 5300 of this request.

5301 **3.13.2.3.7 Report RSSI Measurements Command**

5302 This command is sent by a device to report its measurements of the link between itself and one or more neighbors. In  
5303 a centralized location scenario, the device that sends this command is the device that needs to be localized.

5304 **3.13.2.3.7.1 Payload Format**

5305 The Report RSSI measurement command SHALL be formatted as in Figure 3-60.

5306 **Figure 3-60. Format of the Report RSSI Measurements Command Payload**

<b>Octets</b>	8	1	N
<b>Data Type</b>	EUI64	uint8	Variable [E1]
<b>Field Name</b>	Measuring Device	N Neighbors	NeighborsInfo

5307  
5308 NeighborsInfo structure is reported in Figure 3-61.

5309 **Figure 3-61. Neighbor Info Structure**

<b>Octets</b>	8	2	2	2	1	1
<b>Data Type</b>	EUI64	int16	int16	int16	int8	uint8
<b>Field Name</b>	Neighbor	X	Y	Z	RSSI	NumberRSSI Measurements

5310  
5311 The fields in the payload have the following meanings:

5312 N Neighbors: Numbers of one-hop neighbours that reported the RSSI; indicates how many *NeighborsInfo* fields are  
5313 present in the message.

5314 Measuring Device: IEEE address of the device that report the measurements (i.e., the one that started the blast  
5315 procedure)

5316 Neighbors information:

5317 X,Y,Z: Coordinates (if present) of the neighbor

5318 Neighbor: IEEE address of the neighbor used to identify it if coordinates are either not present or not valid

5319 RSSI: RSSI value registered by the neighbor that refer to the radio link between itself and measuring device

5320 NumberRSSIMeasurements: How many packets were considered to give the RSSI value (=1 meaning is that no  
5321 mean is supported)

5322 **3.13.2.3.8 Request Own Location Command**

5323 This command is sent by a node wishing to know its own location and it is sent to the device that performs the  
5324 centralized localization algorithm.

5325 **3.13.2.3.8.1 Payload Format**

5326 The Request Own Location command payload SHALL be formatted as illustrated in Figure 3-62. The only field in the  
 5327 payload contains the IEEE address of the blind node, i.e., the node that wishes to know about its own location.

5328 **Figure 3-62. Format of the Request Own Location Command Payload**

<b>Octets</b>	8
<b>Data Type</b>	EUI64
<b>Field Name</b>	IEEE Address of the Blind Node

5329

5330 **3.13.2.3.8.2 Effect On Receipt**

5331 The node receiving the Request Own Location command will then reply with a Set Absolute Location command,  
 5332 telling the requesting entity its location.

5333 **3.13.3 Client**

5334 The client has no cluster specific attributes. The client generates the cluster-specific commands received by the server  
 5335 (see 3.13.2.2), as required by the application. The client receives the cluster-specific commands generated by the  
 5336 server (see 3.13.2.3).

5337 **3.14 Input, Output and Value Clusters**

---

5338 **3.14.1 Overview**

5339 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 5340 etc.

5341 This section specifies a number of clusters which are based on ‘Basic’ properties of the Input, Output and Value  
 5342 objects specified by BACnet (see [A1]).

5343 The clusters specified herein are for use typically in Commercial Building applications, but MAY be used in any  
 5344 application domain.

5345 **3.14.2 Analog Input (Basic)**

5346 The Analog Input (Basic) cluster provides an interface for reading the value of an analog measurement and accessing  
 5347 various characteristics of that measurement. The cluster is typically used to implement a sensor that measures an  
 5348 analog physical quantity.

5349 **3.14.2.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5350 **3.14.2.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AI	Type 2 (server to client)

5351 **3.14.2.3 Cluster Identifiers**

Identifier	Name
0x000c	Analog Input

5352 **3.14.2.4 Server**5353 **3.14.2.4.1 Attributes**

5354 The attributes of this cluster are detailed in Table 3-78.

5355 **Table 3-78. Attributes of the Analog Input (Basic) Server Cluster**

ID	Name	Type	Range	Acc	Default	M/O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x0041	<i>MaxPresentValue</i>	single	-	R*W	-	O
0x0045	<i>MinPresentValue</i>	single	-	R*W	-	O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	single	-	RWP	-	M
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x006A	<i>Resolution</i>	single	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	RP	0	M
0x0075	<i>EngineeringUnits</i>	enum1 6	See section 3.14.11.10	R*W	-	O
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5356

5357 For an explanation of the attributes, see section 3.14.11.

5358 **3.14.2.5 Commands**

5359 No cluster specific commands are received or generated.



5360 **3.14.2.6 Attribute Reporting**

5361 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
 5362 according to the minimum and maximum reporting interval, value and timeout settings.

5363 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

5364 **3.14.2.7 Client**

5365 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
 5366 specific commands.

5367 **3.14.3 Analog Output (Basic)**

5368 The Analog Output (Basic) cluster provides an interface for setting the value of an analog output (typically to the  
 5369 environment) and accessing various characteristics of that value.

5370 **3.14.3.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5371 **3.14.3.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AO	Type 2 (server to client)

5372 **3.14.3.3 Cluster Identifiers**

Identifier	Name
0x000d	Analog Output

5373 **3.14.3.4 Server**

5374 **3.14.3.4.1 Attributes**

5375 The attributes of this cluster are detailed in Table 3-79.

5376 **Table 3-79. Attributes of the Analog Output (Basic) Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x0041	<i>MaxPresentValue</i>	single	-	R*W	-	O
0x0045	<i>MinPresentValue</i>	single	-	R*W	-	O

Identifier	Name	Type	Range	Access	Default	M/O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	single	-	RWP	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, single)	-	RW	16 x (0, 0.0)	O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	single	-	R*W	-	O
0x006A	<i>Resolution</i>	single	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	RP	0	M
0x0075	<i>EngineeringUnits</i>	enum16	See section 3.14.11.10	R*W	-	O
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5377

5378 For an explanation of the attributes, see section 3.14.11.

### 5379 3.14.3.5 Commands

5380 No cluster specific commands are received or generated.

### 5381 3.14.3.6 Attribute Reporting

5382 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
5383 according to the minimum and maximum reporting interval, value and timeout settings.5384 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

### 5385 3.14.3.7 Client

5386 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
5387 specific commands.

## 5388 3.14.4 Analog Value (Basic)

5389 The Analog Value (Basic) cluster provides an interface for setting an analog value, typically used as a control system  
5390 parameter, and accessing various characteristics of that value.

5391 **3.14.4.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5392 **3.14.4.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AV	Type 2 (server to client)

5393 **3.14.4.3 Cluster Identifiers**

Identifier	Name
0x000e	Analog Value

5394 **3.14.4.4 Server**

5395 **3.14.4.4.1 Attributes**

5396 The attributes of this cluster are detailed in Table 3-80.

5397 **Table 3-80. Attributes of the Analog Value (Basic) Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	single	-	R/W	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, single)	-	R/W	16 x (0, 0.0)	O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	single	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0075	<i>EngineeringUnits</i>	enum16	See section 3.14.11.10	R*W	-	O
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5398

5399 For an explanation of the attributes, see section 3.14.11.

5400 **3.14.4.5 Commands**

5401 No cluster specific commands are received or generated.

5402 **3.14.4.6 Attribute Reporting**5403 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
5404 according to the minimum and maximum reporting interval, value and timeout settings.5405 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*5406 **3.14.4.7 Client**5407 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
5408 specific commands.5409 **3.14.5 Binary Input (Basic)**5410 The Binary Input (Basic) cluster provides an interface for reading the value of a binary measurement and accessing  
5411 various characteristics of that measurement. The cluster is typically used to implement a sensor that measures a two-  
5412 state physical quantity.5413 **3.14.5.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5414 **3.14.5.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BI	Type 2 (server to client)

5415 **3.14.5.3 Cluster Identifiers**

Identifier	Name
0x000f	Binary Input

5416 **3.14.5.4 Server**5417 **3.14.5.4.1 Attributes**

5418 The attributes of this cluster are detailed in Table 3-81.

5419

**Table 3-81. Attributes of the Binary Input (Basic) Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0004	<i>ActiveText</i>	string	-	R*W	Null string	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x002E	<i>InactiveText</i>	string	-	R*W	Null string	O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0054	<i>Polarity</i>	enum8	-	R	0	O
0x0055	<i>PresentValue</i>	bool	-	R*W	-	M
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5420

5421 For an explanation of the attributes, see section 3.14.11.

### 5422 **3.14.5.5 Commands**

5423 No cluster specific commands are received or generated.

### 5424 **3.14.5.6 Attribute Reporting**

5425 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
 5426 according to the minimum and maximum reporting interval, value and timeout settings.

5427 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

### 5428 **3.14.5.7 Client**

5429 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
 5430 specific commands.

## 5431 **3.14.6 Binary Output (Basic)**

5432 The Binary Output (Basic) cluster provides an interface for setting the value of a binary output, and accessing various  
 5433 characteristics of that value.

### 5434 **3.14.6.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5435 **3.14.6.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BO	Type 2 (server to client)

5436 **3.14.6.3 Cluster Identifiers**

Identifier	Name
0x0010	Binary Output

5437 **3.14.6.4 Server**5438 **3.14.6.4.1 Attributes**

5439 The attributes of this cluster are detailed in Table 3-82.

5440 **Table 3-82. Attributes of the Binary Output (Basic) Server Cluster**

ID	Name	Type	Range	Access	Default	MO
0x0004	<i>ActiveText</i>	string	-	R*W	Null string	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x002E	<i>InactiveText</i>	string	-	R*W	Null string	O
0x0042	<i>MinimumOffTime</i>	uint32	-	R*W	0xffffffff	O
0x0043	<i>MinimumOnTime</i>	uint32	-	R*W	0xffffffff	O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0054	<i>Polarity</i>	enum8	-	R	0	O
0x0055	<i>PresentValue</i>	bool	-	R*W	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, bool)	-	R/W	16 x (0, 0)	O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	bool	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5441 For an explanation of the attributes, see section 3.14.11.

5442 **3.14.6.5 Commands**

5443 No cluster specific commands are received or generated.

5444 **3.14.6.6 Attribute Reporting**

5445 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
 5446 according to the minimum and maximum reporting interval, value and timeout settings.

5447 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

5448 **3.14.6.7 Client**

5449 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

5450 **3.14.7 Binary Value (Basic)**

5451 The Binary Value (Basic) cluster provides an interface for setting a binary value, typically used as a control system  
 5452 parameter, and accessing various characteristics of that value.

5453 **3.14.7.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5454 **3.14.7.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BV	Type 2 (server to client)

5455 **3.14.7.3 Cluster Identifiers**

Identifier	Name
0x0011	Binary Value

5456 **3.14.7.4 Server**

5457 **3.14.7.4.1 Attributes**

5458 The attributes of this cluster are detailed in Table 3-83.

5459 **Table 3-83. Attributes of the Binary Value (Basic) Server Cluster**

ID	Name	Type	Range	Access	Default	M/O
0x0004	<i>ActiveText</i>	string	-	R*W	Null string	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O

ID	Name	Type	Range	Access	Default	M/O
0x002E	<i>InactiveText</i>	string	-	R*W	Null string	O
0x0042	<i>MinimumOffTime</i>	uint32	-	R*W	0xffffffff	O
0x0043	<i>MinimumOnTime</i>	uint32	-	R*W	0xffffffff	O
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	bool	-	R*W	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, bool)	-	R/W	16 x (0, 0)	O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	bool	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5460 For an explanation of the attributes, see section 3.14.11.

### 5461 3.14.7.5 Commands

5462 No cluster specific commands are received or generated.

### 5463 3.14.7.6 Attribute Reporting

5464 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
5465 according to the minimum and maximum reporting interval, value and timeout settings.

5466 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

### 5467 3.14.7.7 Client

5468 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
5469 specific commands.

## 5470 3.14.8 Multistate Input (Basic)

5471 The Multistate Input (Basic) cluster provides an interface for reading the value of a multistate measurement and  
5472 accessing various characteristics of that measurement. The cluster is typically used to implement a sensor that  
5473 measures a physical quantity that can take on one of a number of discrete states.

### 5474 3.14.8.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added



5475 **3.14.8.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MI	Type 2 (server to client)

5476 **3.14.8.3 Cluster Identifiers**

Identifier	Name
0x0012	Multistate Input

5477 **3.14.8.4 Server**

5478 **3.14.8.4.1 Attributes**

5479 The attributes of this cluster are detailed in Table 3-84.

5480 **Table 3-84. Attributes of the Multistate Input (Basic) Server Cluster**

Identifier	Name	Type	Range	Acc	Default	MO
0x000E	<i>StateText</i>	Array of character string	-	R*W	Null	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x004A	<i>NumberOfStates</i>	uint16	1 to 0xffff	R*W	0	M
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	uint16	-	R*W	-	M
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5481 For an explanation of the attributes, see section 3.14.11.

5482 **3.14.8.5 Commands**

5483 No cluster specific commands are received or generated.

5484 **3.14.8.6 Attribute Reporting**

5485 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
 5486 according to the minimum and maximum reporting interval, value and timeout settings.

5487 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

5488 **3.14.8.7 Client**

5489 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
5490 specific commands.

5491 **3.14.9 Multistate Output (Basic)**

5492 The Multistate Output (Basic) cluster provides an interface for setting the value of an output that can take one of a  
5493 number of discrete values, and accessing characteristics of that value.

5494 **3.14.9.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

5495 **3.14.9.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MO	Type 2 (server to client)

5496 **3.14.9.3 Cluster Identifiers**

Identifier	Name
0x0013	Multistate Output

5497 **3.14.9.4 Server**5498 **3.14.9.4.1 Attributes**

5499 The attributes of this cluster are detailed in Table 3-85.

5500 **Table 3-85. Attributes of the Multistate Output (Basic) Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x000E	<i>StateText</i>	Array of character string	-	R*W	Null	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x004A	<i>NumberOfStates</i>	uint16	1 to 0xffff	R*W	0	M
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	uint16	-	R/W	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, uint16)	-	R/W	16 x (0, 0)	O

Identifier	Name	Type	Range	Access	Default	M/O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	uint16	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5501 For an explanation of the attributes, see section 3.14.11.

### 5502 3.14.9.5 Commands

5503 No cluster specific commands are received or generated.

### 5504 3.14.9.6 Attribute Reporting

5505 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands,  
 5506 according to the minimum and maximum reporting interval, value and timeout settings.

5507 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*

### 5508 3.14.9.7 Client

5509 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
 5510 specific commands.

## 5511 3.14.10 Multistate Value (Basic)

5512 The Multistate Value (Basic) cluster provides an interface for setting a multistate value, typically used as a control  
 5513 system parameter, and accessing characteristics of that value.

### 5514 3.14.10.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 5515 3.14.10.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MV	Type 2 (server to client)

### 5516 3.14.10.3 Cluster Identifiers

Identifier	Name
0x0014	Multistate Value

5517 **3.14.10.4 Server**5518 **3.14.10.4.1 Attributes**

5519 The attributes of this cluster are detailed in Table 3-86.

5520 **Table 3-86. Attributes of the Multistate Value (Basic) Server Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x000E	<i>StateText</i>	Array of character string	-	R*W	Null	O
0x001C	<i>Description</i>	string	-	R*W	Null string	O
0x004A	<i>NumberOfStates</i>	uint16	1 to 0xffff	R*W	0	M
0x0051	<i>OutOfService</i>	bool	False (0) or True (1)	R*W	False (0)	M
0x0055	<i>PresentValue</i>	uint16	-	R/W	-	M
0x0057	<i>PriorityArray</i>	Array of 16 structures of (bool, uint16)	-	R/W	16 x (0, 0)	O
0x0067	<i>Reliability</i>	enum8	-	R*W	0x00	O
0x0068	<i>RelinquishDefault</i>	uint16	-	R*W	-	O
0x006F	<i>StatusFlags</i>	map8	0x00 to 0x0f	R	0	M
0x0100	<i>ApplicationType</i>	uint32	0 to 0xffffffff	R	-	O

5521 For an explanation of the attributes, see section 3.14.11.

5522 **3.14.10.5 Commands**

5523 No cluster specific commands are received or generated.

5524 **3.14.10.6 Attribute Reporting**

5525 This cluster SHALL support attribute reporting using the Report Attributes and Configure Reporting commands, according to the minimum and maximum reporting interval, value and timeout settings.

5527 The following attributes SHALL be reported: *StatusFlags*, *PresentValue*5528 **3.14.10.7 Client**

5529 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster specific commands.

5531 **3.14.11 Attribute Descriptions**

5532 Note: These attributes are based on BACnet properties with the same names. For more information, refer to the BACnet reference manual [A1].

5533

### 5534 3.14.11.1 OutOfService Attribute

5535 The *OutOfService* attribute, of type Boolean, indicates whether (TRUE) or not (FALSE) the physical input, output or  
5536 value that the cluster represents is not in service. For an Input cluster, when *OutOfService* is TRUE the *PresentValue*  
5537 attribute is decoupled from the physical input and will not track changes to the physical input. For an Output cluster,  
5538 when *OutOfService* is TRUE the *PresentValue* attribute is decoupled from the physical output, so changes to  
5539 *PresentValue* will not affect the physical output. For a Value cluster, when *OutOfService* is TRUE the *PresentValue*  
5540 attribute MAY be written to freely by software local to the device that the cluster resides on.

### 5541 3.14.11.2 PresentValue Attribute

5542 The *PresentValue* attribute indicates the current value of the input, output or value, as appropriate for the cluster. For  
5543 Analog clusters it is of type single precision, for Binary clusters it is of type Boolean, and for multistate clusters it is  
5544 of type Unsigned 16-bit integer.

5545 The *PresentValue* attribute of an input cluster SHALL be writable when *OutOfService* is TRUE.

5546 When the *PriorityArray* attribute is implemented, writing to *PresentValue* SHALL be equivalent to writing to element  
5547 16 of *PriorityArray*, i.e., with a priority of 16.

### 5548 3.14.11.3 StatusFlags Attribute

5549 This attribute, of type bitmap, represents four Boolean flags that indicate the general “health” of the analog sensor.  
5550 Three of the flags are associated with the values of other optional attributes of this cluster. A more detailed status  
5551 could be determined by reading the optional attributes (if supported) that are linked to these flags. The relationship  
5552 between individual flags is not defined. The four flags are

5553 Bit 0 = IN ALARM, Bit 1 = FAULT, Bit 2 = OVERRIDDEN, Bit 3 = OUT OF SERVICE

5554 where:

5555 IN ALARM - Logical FALSE (0) if the *EventState* attribute has a value of NORMAL, otherwise logical TRUE (1).  
5556 This bit is always 0 unless the cluster implementing the *EventState* attribute is implemented on the same endpoint.

5557 FAULT - Logical TRUE (1) if the *Reliability* attribute is present and does not have a value of NO FAULT  
5558 DETECTED, otherwise logical FALSE (0).

5559 OVERRIDDEN - Logical TRUE (1) if the cluster has been overridden by some mechanism local to the device.  
5560 Otherwise, the value is logical FALSE (0).

5561 In this context, for an input cluster, “overridden” is taken to mean that the *PresentValue* and *Reliability* (optional)  
5562 attributes are no longer tracking changes to the physical input. For an Output cluster, “overridden” is taken to mean  
5563 that the physical output is no longer tracking changes to the *PresentValue* attribute and the *Reliability* attribute is no  
5564 longer a reflection of the physical output. For a Value cluster, “overridden” is taken to mean that the *PresentValue*  
5565 attribute is not writeable.

5566 OUT OF SERVICE - Logical TRUE (1) if the *OutOfService* attribute has a value of TRUE, otherwise logical FALSE  
5567 (0).

### 5568 3.14.11.4 Description Attribute

5569 The *Description* attribute, of type Character string, MAY be used to hold a description of the usage of the input, output  
5570 or value, as appropriate to the cluster. The character set used SHALL be ASCII, and the attribute SHALL contain a  
5571 maximum of 16 characters, which SHALL be printable but are otherwise unrestricted.

### 5572 3.14.11.5 MaxPresentValue Attribute

5573 The *MaxPresentValue* attribute, of type Single precision, indicates the highest value that can be reliably obtained for  
5574 the *PresentValue* attribute of an Analog Input cluster, or which can reliably be used for the *PresentValue* attribute of  
5575 an Analog Output or Analog Value cluster.

### 5576 3.14.11.6 PriorityArray Attribute

5577 The *PriorityArray* attribute is an array of 16 structures. The first element of each structure is a Boolean, and the second  
5578 element is of the same type as the *PresentValue* attribute of the corresponding cluster.

5579 *PriorityArray* holds potential values for the *PresentValue* attribute of the corresponding cluster, in order of decreasing  
5580 priority. The first value in the array corresponds to priority 1 (highest), the second value corresponds to priority 2, and  
5581 so on, to the sixteenth value that corresponds to priority 16 (lowest).

5582 The Boolean value in each element of the array indicates whether (TRUE) or not (FALSE) there is a valid value at  
5583 that priority. All entries within the priority table are continuously monitored in order to locate the entry with the highest  
5584 priority valid value, and *PresentValue* is set to this value.

5585 When *PriorityArray* is supported, *PresentValue* MAY be written to indirectly by writing to the *PriorityArray*, as  
5586 described above. If *PresentValue* is written to directly, a default priority of 16 (the lowest priority) SHALL be  
5587 assumed, and the value is entered into the 16th element of *PriorityArray*.

5588 When a value at a given priority is marked as invalid, by writing FALSE to its corresponding Boolean value, it is said  
5589 to be relinquished.

5590 **(Informative note:** In BACnet, each element of *PriorityArray* consists of a single value, which MAY be either of the  
5591 same type as *PresentValue* or MAY be of type NULL to indicate that a value is not present. An attribute cannot have  
5592 a variable data type; thus, an extra Boolean value is associated with each element of the array to indicate whether or  
5593 not it is null).

### 5594 3.14.11.7 RelinquishDefault Attribute

5595 The *RelinquishDefault* attribute is the default value to be used for the *PresentValue* attribute when all elements of the  
5596 *PriorityArray* attribute are marked as invalid.

### 5597 3.14.11.8 MinPresentValue Attribute

5598 The *MinPresentValue* attribute, of type Single precision, indicates the lowest value that can be reliably obtained for  
5599 the *PresentValue* attribute of an Analog Input cluster, or which can reliably be used for the *PresentValue* attribute of  
5600 an Analog Output or Analog Value cluster.

### 5601 3.14.11.9 Reliability Attribute

5602 The *Reliability* attribute, of type 8-bit enumeration, provides an indication of whether the *PresentValue* or the  
5603 operation of the physical input, output or value in question (as appropriate for the cluster) is “reliable” as far as can be  
5604 determined and, if not, why not. The *Reliability* attribute MAY have any of the following values:

5605 NO-FAULT-DETECTED (0)

5606 NO-SENSOR (1) - for input clusters only

5607 OVER-RANGE (2)

5608 UNDER-RANGE (3)

5609 OPEN-LOOP (4)

5610 SHORTED-LOOP (5)

- 5611 NO-OUTPUT (6) - for input clusters only
- 5612 UNRELIABLE-OTHER (7)
- 5613 PROCESS-ERROR (8)
- 5614 MULTI-STATE-FAULT (9) - for multistate clusters only
- 5615 CONFIGURATION-ERROR (10)

#### 5616 **3.14.11.10 EngineeringUnits Attribute**

- 5617 The *EngineeringUnits* attribute indicates the physical units associated with the value of the *PresentValue* attribute of
- 5618 an Analog cluster.
- 5619 Values 0x0000 to 0x00fe are reserved for the list of engineering units with corresponding values specified in Clause
- 5620 21 of the BACnet standard [A1]. 0x00ff represents 'other'. Values 0x0100 to 0xffff are available for proprietary use.
- 5621 If the *ApplicationType* attribute is implemented, and is set to a value with a defined physical unit, the physical unit
- 5622 defined in *ApplicationType* takes priority over *EngineeringUnits*.
- 5623 This attribute is defined to be Read Only, but a vendor can decide to allow this to be written to if *ApplicationType* is
- 5624 also supported. If this attribute is written to, how the device handles invalid units (e.g., changing Deg F to Cubic Feet
- 5625 per Minute), any local display or other vendor-specific operation (upon the change) is a local matter.

#### 5626 **3.14.11.11 Resolution Attribute**

- 5627 This attribute, of type Single precision, indicates the smallest recognizable change to *PresentValue*.

#### 5628 **3.14.11.12 ActiveText Attribute**

- 5629 This attribute, of type Character string, MAY be used to hold a human readable description of the ACTIVE state of a
- 5630 binary *PresentValue*. For example, for a Binary Input cluster, if the physical input is a switch contact, then the
- 5631 *ActiveText* attribute might be assigned a value such as “Fan 1 On”. If either the *ActiveText* attribute or the *InactiveText*
- 5632 attribute are present, then both of them SHALL be present.
- 5633 The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which
- 5634 SHALL be printable but are otherwise unrestricted.

#### 5635 **3.14.11.13 InactiveText Attribute**

- 5636 This attribute, of type Character string, MAY be used to hold a human readable description of the INACTIVE state
- 5637 of a binary *PresentValue*. For example, for a Binary Input cluster, if the physical input is a switch contact, then the
- 5638 *InactiveText* attribute might be assigned a value such as “Fan 1 Off”. If either the *InactiveText* attribute or the
- 5639 *ActiveText* attribute are present, then both of them SHALL be present.
- 5640 The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which
- 5641 SHALL be printable but are otherwise unrestricted.

#### 5642 **3.14.11.14 MinimumOffTime Attribute**

- 5643 This property, of type 32-bit unsigned integer, represents the minimum number of seconds that a binary *PresentValue*
- 5644 SHALL remain in the INACTIVE (0) state after a write to *PresentValue* causes it to assume the INACTIVE state.

**5645 3.14.11.15 MinimumOnTime Attribute**

5646 This property, of type 32-bit unsigned integer, represents the minimum number of seconds that a binary *PresentValue*  
5647 SHALL remain in the ACTIVE (1) state after a write to *PresentValue* causes it to assume the ACTIVE state.

**5648 3.14.11.16 Polarity Attribute**

5649 This attribute, of type enumeration, indicates the relationship between the physical state of the input (or output as  
5650 appropriate for the cluster) and the logical state represented by a binary *PresentValue* attribute, when *OutOfService* is  
5651 FALSE. If the *Polarity* attribute is NORMAL (0), then the ACTIVE (1) state of the *PresentValue* attribute is also the  
5652 ACTIVE or ON state of the physical input (or output). If the *Polarity* attribute is REVERSE (1), then the ACTIVE  
5653 (1) state of the *PresentValue* attribute is the INACTIVE or OFF state of the physical input (or output).

5654 Thus, when *OutOfService* is FALSE, for a constant physical input state a change in the *Polarity* attribute SHALL  
5655 produce a change in the *PresentValue* attribute. If *OutOfService* is TRUE, then the *Polarity* attribute SHALL have no  
5656 effect on the *PresentValue* attribute.

**5657 3.14.11.17 NumberOfStates Attribute**

5658 This attribute, of type Unsigned 16-bit integer, defines the number of states that a multistate *PresentValue* MAY have.  
5659 The *NumberOfStates* property SHALL always have a value greater than zero. If the value of this property is changed,  
5660 the size of the *StateText* array, if present, SHALL also be changed to the same value. The states are numbered  
5661 consecutively, starting with 1.

**5662 3.14.11.18 StateText Attribute**

5663 This attribute, of type Array of Character strings, holds descriptions of all possible states of a multistate *PresentValue*.  
5664 The number of descriptions matches the number of states defined in the *NumberOfStates* property. The *PresentValue*,  
5665 interpreted as an integer, serves as an index into the array. If the size of this array is changed, the *NumberOfStates*  
5666 property SHALL also be changed to the same value.

5667 The character set used SHALL be ASCII, and the attribute SHALL contain a maximum of 16 characters, which  
5668 SHALL be printable but are otherwise unrestricted.

**5669 3.14.11.19 ApplicationType Attribute**

5670 The *ApplicationType* attribute is an unsigned 32 bit integer that indicates the specific application usage for this cluster.  
5671 (**Note:** This attribute has no BACnet equivalent.)

5672 *ApplicationType* is subdivided into Group, Type and an Index number, as follows.

5673	Group	=	Bits	24	to	31
5674	An indication of the cluster this attribute is part of.					
5675	Type	=	Bits	16	to	23
5676	For Analog clusters, the physical quantity that the Present Value attribute of the cluster represents.					
5677	For Binary and Multistate clusters, the application usage domain.					
5678	Index	=	Bits	0	to	15
5679	The specific application usage of the cluster.					

**5680 3.14.11.19.1 Analog Input (AI) Types**

5681 Group = 0x00.

5682 The following sub-clauses describe the values when Type = 0x00 to 0x0E. Types 0x0F to 0xFE are reserved, Type =  
5683 0xFF indicates other.



5684 **3.14.11.19.1.1 Type = 0x00: Temperature in degrees C**

5685 **Table 3-87. AI Types, Type = 0x00: Temperature in Degrees C**

Index	Application Usage
0x0000	2 Pipe Entering Water Temperature AI
0x0001	2 Pipe Leaving Water Temperature AI
0x0002	Boiler Entering Temperature AI
0x0003	Boiler Leaving Temperature AI
0x0004	Chiller Chilled Water Entering Temp AI
0x0005	Chiller Chilled Water Leaving Temp AI
0x0006	Chiller Condenser Water Entering Temp AI
0x0007	Chiller Condenser Water Leaving Temp AI
0x0008	Cold Deck Temperature AI
0x0009	Cooling Coil Discharge Temperature AI
0x000A	Cooling Entering Water Temperature AI
0x000B	Cooling Leaving Water Temperature AI
0x000C	Condenser Water Return Temperature AI
0x000D	Condenser Water Supply Temperature AI
0x000E	Decouple Loop Temperature AI
0x000F	Building Load AI
0x0010	Decouple Loop Temperature AI
0x0011	Dew Point Temperature AI
0x0012	Discharge Air Temperature AI
0x0013	Discharge Temperature AI
0x0014	Exhaust Air Temperature After Heat Recovery AI
0x0015	Exhaust Air Temperature AI
0x0016	Glycol Temperature AI
0x0017	Heat Recovery Air Temperature AI
0x0018	Hot Deck Temperature AI

<b>Index</b>	<b>Application Usage</b>
0x0019	Heat Exchanger Bypass Temp AI
0x001A	Heat Exchanger Entering Temp AI
0x001B	Heat Exchanger Leaving Temp AI
0x001C	Mechanical Room Temperature AI
0x001D	Mixed Air Temperature AI
0x001E	Mixed Air Temperature AI
0x001F	Outdoor Air Dewpoint Temp AI
0x0020	Outdoor Air Temperature AI
0x0021	Preheat Air Temperature AI
0x0022	Preheat Entering Water Temperature AI
0x0023	Preheat Leaving Water Temperature AI
0x0024	Primary Chilled Water Return Temp AI
0x0025	Primary Chilled Water Supply Temp AI
0x0026	Primary Hot Water Return Temp AI
0x0027	Primary Hot Water Supply Temp AI
0x0028	Reheat Coil Discharge Temperature AI
0x0029	Reheat Entering Water Temperature AI
0x002A	Reheat Leaving Water Temperature AI
0x002B	Return Air Temperature AI
0x002C	Secondary Chilled Water Return Temp AI
0x002D	Secondary Chilled Water Supply Temp AI
0x002E	Secondary HW Return Temp AI
0x002F	Secondary HW Supply Temp AI
0x0030	Sideloop Reset Temperature AI
0x0031	Sideloop Temperature Setpoint AI
0x0032	Sideloop Temperature AI
0x0033	Source Temperature

Index	Application Usage
0x0034	Supply Air Temperature AI
0x0035	Supply Low Limit Temperature AI
0x0036	Tower Basin Temp AI
0x0037	Two Pipe Leaving Water Temp AI
0x0038	Reserved
0x0039	Zone Dewpoint Temperature AI
0x003A	Zone Sensor Setpoint AI
0x003B	Zone Sensor Setpoint Offset AI
0x003C	Zone Temperature AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5686 **3.14.11.19.1.2 Type = 0x01: Relative Humidity in %**

5687 **Table 3-88. AI Types, Type = 0x01: Relative Humidity in %**

Index	Application Usage
0x0000	Discharge Humidity AI
0x0001	Exhaust Humidity AI
0x0002	Hot Deck Humidity AI
0x0003	Mixed Air Humidity AI
0x0004	Outdoor Air Humidity AI
0x0005	Return Humidity AI
0x0006	Sideloop Humidity AI
0x0007	Space Humidity AI
0x0008	Zone Humidity AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5688 **3.14.11.19.1.3 Type = 0x02: Pressure in Pascal**

5689

**Table 3-89. AI Types, Type = 0x02: Pressure in Pascal**

<b>Index</b>	<b>Application Usage</b>
0x0000	Boiler Pump Differential Pressure AI
0x0001	Building Static Pressure AI
0x0002	Cold Deck Differential Pressure Sensor AI
0x0003	Chilled Water Building Differential Pressure AI
0x0004	Cold Deck Differential Pressure AI
0x0005	Cold Deck Static Pressure AI
0x0006	Condenser Water Pump Differential Pressure AI
0x0007	Discharge Differential Pressure AI
0x0008	Discharge Static Pressure 1 AI
0x0009	Discharge Static Pressure 2 AI
0x000A	Exhaust Air Differential Pressure AI
0x000B	Exhaust Air Static Pressure AI
0x000C	Exhaust Differential Pressure AI
0x000D	Exhaust Differential Pressure AI
0x000E	Hot Deck Differential Pressure AI
0x000F	Hot Deck Differential Pressure AI
0x0010	Hot Deck Static Pressure AI
0x0011	Hot Water Bldg Diff Pressure AI
0x0012	Heat Exchanger Steam Pressure AI
0x0013	Minimum Outdoor Air Differential Pressure AI
0x0014	Outdoor Air Differential Pressure AI
0x0015	Primary Chilled Water Pump Differential Pressure AI
0x0016	Primary Hot water Pump Differential Pressure AI
0x0017	Relief Differential Pressure AI
0x0018	Return Air Static Pressure AI
0x0019	Return Differential Pressure AI

Index	Application Usage
0x001A	Secondary Chilled Water Pump Differential Pressure AI
0x001B	Secondary Hot water Pump Differential Pressure AI
0x001C	Sideloop Pressure AI
0x001D	Steam Pressure AI
0x001E	Supply Differential Pressure Sensor AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5690 **3.14.11.19.1.4 Type = 0x03: Flow in Liters/Second**

5691 **Table 3-90. AI Types, Type = 0x03: Flow in Liters/Second**

Index	Application Usage
0x0000	Chilled Water Flow AI
0x0001	Chiller Chilled Water Flow AI
0x0002	Chiller Condenser Water Flow AI
0x0003	Cold Deck Flow AI
0x0004	Decouple Loop Flow AI
0x0005	Discharge Flow AI
0x0006	Exhaust Fan Flow AI
0x0007	Exhaust Flow AI
0x0008	Fan Flow AI
0x0009	Hot Deck Flow AI
0x000A	Hot Water Flow AI
0x000B	Minimum Outdoor Air Fan Flow AI
0x000C	Minimum Outdoor Air Flow AI
0x000D	Outdoor Air Flow AI
0x000E	Primary Chilled Water Flow AI
0X000F	Relief Fan Flow AI

Index	Application Usage
0x0010	Relief Flow AI
0x0011	Return Fan Flow AI
0x0012	Return Flow AI
0x0013	Secondary Chilled Water Flow AI
0x0014	Supply Fan Flow AI
0x0015	Tower Fan Flow AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5692 **3.14.11.19.1.5 Type = 0x04: Percentage %**

5693 **Table 3-91. AI Types, Type = 0x04: Percentage %**

Index	Application Usage
0x0000	Chiller % Full Load Amperage AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5694 **3.14.11.19.1.6 Type = 0x05: Parts per Million PPM**

5695 **Table 3-92. AI types, Type = 0x05: Parts per Million PPM**

Index	Application Usage
0x0000	Return Carbon Dioxide AI
0x0001	Zone Carbon Dioxide AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5696 **3.14.11.19.1.7 Type = 0x06: Rotational Speed in RPM**

5697

**Table 3-93. AI Types, Type = 0x06: Rotational Speed in RPM**

Index	Application Usage
0x0000	Exhaust Fan Remote Speed AI
0x0001	Heat Recovery Wheel Remote Speed AI
0x0002	Min Outdoor Air Fan Remote Speed AI
0x0003	Relief Fan Remote Speed AI
0x0004	Return Fan Remote Speed AI
0x0005	Supply Fan Remote Speed AI
0x0006	Variable Speed Drive Motor Speed AI
0x0007	Variable Speed Drive Speed Setpoint AI
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5698 **3.14.11.19.1.8 Type = 0x07: Current in Amps**

5699

**Table 3-94. AI Types, Type = 0x07: Current in Amps**

Index	Application Usage
0x0000	Chiller Amps AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5700 **3.14.11.19.1.9 Type = 0x08: Frequency in Hz**

5701

**Table 3-95. AI Types, Type = 0x08: Frequency in Hz**

Index	Application Usage
0x0000	Variable Speed Drive Output Frequency AI
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5702 **3.14.11.19.1.10 Type = 0x09: Power in Watts**

5703

**Table 3-96. AI Types, Type = 0x09: Power in Watts**

Index	Application Usage
0x0000	Power Consumption AI
0x0200- FFFE	Vendor defined
0xFFFF	Other

5704 **3.14.11.19.1.11 Type = 0x0A: Power in kW**

5705

**Table 3-97. AI Types, Type = 0x0A: Power in kW**

Index	Application Usage
0x0000	Absolute Power AI
0x0001	Power Consumption AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5706 **3.14.11.19.1.12 Type = 0x0B: Energy in kWh**

5707

**Table 3-98. AI Types, Type = 0x0B: Energy in kWh**

Index	Application Usage
0x0000	Variable Speed Drive Kilowatt Hours AI
0x0200- FFFE	Vendor defined
0xFFFF	Other

5708 **3.14.11.19.1.13 Type = 0x0C: Count – Unitless**

5709

**Table 3-99. AI Types, Type = 0x0C: Count - Unitless**

Index	Application Usage
0x0000	Count
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5710 **3.14.11.19.1.14 Type = 0x0D: Enthalpy in KJoules/Kg**



5711

**Table 3-100. AI Types, Type = 0x0D: Enthalpy in KJoules/Kg**

Index	Application Usage
0x0000	Outdoor Air Enthalpy AI
0x0001	Return Air Enthalpy AI
0x0002	Space Enthalpy
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5712 **3.14.11.19.1.15 Type = 0x0E: Time in Seconds**

5713

**Table 3-101. AI types, Type = 0x0E: Time in Seconds**

Index	Application Usage
0x0000	Relative time AI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5714 **3.14.11.19.2 Analog Output (AO) types**

5715 Group = 0x01.

5716 The following sub-clauses describe the values when Type = 0x00 to 0x0E. Types 0x0F to 0xFE are reserved, Type =  
 5717 0xFF indicates other.

5718 **3.14.11.19.2.1 Type = 0x00: Temperature in Degrees C**

5719

**Table 3-102. AO Types, Type = 0x00: Temperature in Degrees C**

Index	Application Usage
0x0000	Boiler AO
0x0001	Boiler Setpoint AO
0x0002	Cold Deck AO
0x0003	Chiller Setpoint AO
0x0004	Chiller Setpoint AO
0x0005	Hot Deck AO

Index	Application Usage
0x0006	Cooling Valve AO
0x0007	Zone Temperature Setpoint AO
0x0008	Setpoint Offset AO
0x0009	Setpoint Shift AO
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5720 **3.14.11.19.2.2 Type = 0x01: Relative Humidity in %**

5721 **Table 3-103. AO Types, Type = 0x01: Relative Humidity in %**

Index	Application Usage
0x0000	Humidification AO
0x0001	Zone Relative Humidity Setpoint AO
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5722 **3.14.11.19.2.3 Type = 0x02: Pressure Pascal**

5723 **Table 3-104. AO Types, Type = 0x02: Pressure Pascal**

Index	Application Usage
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5724 **3.14.11.19.2.4 Type = 0x03: Flow in Liters/Second**

5725 **Table 3-105. AO Types, Type = 0x03: Flow in Liters/Second**

Index	Application Usage
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5726 **3.14.11.19.2.5 Type = 0x04: Percentage %**

5727

**Table 3-106. AO Types, Type = 0x04: Percentage %**

<b>Index</b>	<b>Application Usage</b>
0x0000	Face & Bypass Damper AO
0x0001	Heat Recovery Valve AO
0x0002	Heat Recovery Wheel AO
0x0003	Heating Valve AO
0x0004	Hot Deck Damper AO
0x0005	2 Pipe Damper AO
0x0006	2 Pipe Valve AO
0x0007	Boiler Mixing Valve AO
0x0008	Box Cooling Valve AO
0x0009	Box Heating Valve AO
0x000A	Chilled Water Bypass Valve AO
0x000B	Cold Deck Damper AO
0x000C	Cooling Damper AO
0x000D	Cooling Valve AO
0x000E	Damper AO
0x000F	Exhaust Air Damper AO
0x0010	Exhaust Damper AO
0x0011	Hot Water Bypass Valve AO
0x0012	Hot Water Mixing Valve AO
0x0013	Minimum Outside Air Damper AO
0x0014	Minimum Outside Air Fan AO
0x0015	Mixed Air Damper AO
0x0016	Mixing Valve AO
0x0017	Outside Air Damper AO

<b>Index</b>	<b>Application Usage</b>
0x0018	Primary Chilled Water Pump AO
0x0019	Primary Hot Water Pump AO
0x001A	Primary Heat Exchange Pump AO
0x001B	Preheat Damper AO
0x001C	Preheat Valve AO
0x001D	Reheat Valve 1 AO
0x001E	Reheat Valve AO
0x001F	Return Air Damper AO
0x0020	Secondary Chilled Water Pump AO
0x0021	Sequenced Valves AO
0x0022	Secondary Hot Water Pump AO
0x0023	Secondary Heat Exchange Pump AO
0x0024	Sideloop AO
0x0025	Supply Heating Valve AO
0x0026	Supply Damper AO
0x0027	Tower Bypass Valve AO
0x0028	Tower Fan AO
0x0029	Valve AO
0x002A	Zone 1 Damper AO
0x002B	Zone 1 Heating Valve AO
0x002C	Heat Recovery Exhaust Bypass Damper AO
0x002D	Heat Recovery Outside Air Bypass Damper AO
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5728 **3.14.11.19.2.6 Type = 0x05: Parts per Million PPM**

5729

**Table 3-107. AO Types, Type = 0x05: Parts per Million PPM**

Index	Application Usage
0x0000	Space Carbon Dioxide limit AO
0x0200 to 0xFFFF	Vendor defined
0xFFFF	Other

5730 **3.14.11.19.2.7 Type = 0x06: Rotational Speed RPM**

5731

**Table 3-108. AO Types, Type = 0x06: Rotational Speed RPM**

Index	Application Usage
0x0000	Exhaust Fan Speed AO
0x0001	Fan Speed AO
0x0002	Relief Fan Speed AO
0x0003	Return Fan Speed AO
0x0004	Supply Fan Speed AO
0x0200- 0xFFFF	Vendor defined
0xFFFF	Other

5732 **3.14.11.19.2.8 Type = 0x07: Current in Amps**

5733

**Table 3-109. AO Types, Type = 0x07: Current in Amps**

Index	Application Usage
0x0200- 0xFFFF	Vendor defined
0xFFFF	Other

5734 **3.14.11.19.2.9 Type = 0x08: Frequency in Hz**

5735

**Table 3-110. AO Types, Type = 0x08: Frequency in Hz**

Index	Application Usage
0x0000 to 0x01FF	Reserved
0x0200- 0xFFFF	Vendor defined

Index	Application Usage
0xFFFF	Other

5736 **3.14.11.19.2.10 Type = 0x09: Power in Watts**5737 **Table 3-111. AO Types, Type = 0x09: Power in Watts**

Index	Application Usage
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5738 **3.14.11.19.2.11 Type = 0x0A: Power in kW**5739 **Table 3-112. AO Types, Type = 0x0A: Power in kW**

Index	Application Usage
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5740 **3.14.11.19.2.12 Type = 0x0B: Energy in kWh**5741 **Table 3-113. AO Types, Type = 0x0B: Energy in kWh**

Index	Application Usage
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5742 **3.14.11.19.2.13 Type = 0x0C: Count – Unitless**5743 **Table 3-114. AO Types, Type = 0x0C: Count - Unitless**

Index	Application Usage
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5744 **3.14.11.19.2.14 Type = 0x0D: Enthalpy in KJoules/Kg**

5745

**Table 3-115. AO Types, Type = 0x0D: Enthalpy in KJoules/Kg**

Index	Application Usage
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5746 **3.14.11.19.2.15 Type = 0x0E: Time in Seconds**

5747

**Table 3-116. AO Types, Type = 0x0E: Time in Seconds**

Index	Application Usage
0x0000	Relative time AO
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5748 **3.14.11.19.3 Analog Value (AV) Types**

5749 Group = 0x02.

5750 The following sub-clauses describe the values when Type = 0x00 to 0x03. Types 0x04 to 0xFE are reserved, Type =  
 5751 0xFF indicates other.

5752 **3.14.11.19.3.1 Type = 0x00: Temperature in Degrees C**

5753

**Table 3-117. AV Types, Type = 0x00: Temperature in Degrees C**

Index	Application Usage
0x0000	Setpoint Offset AV
0x0001	Temp Deadband AV
0x0002	Occupied Heating Setpoint AV
0x0003	Unoccupied Heating Setpoint AV
0x0004	Occupied Cooling Setpoint AV
0x0005	Unoccupied Cooling Setpoint AV
0x0006	Standby Heat Setpoint AV
0x0007	Standby Cooling Setpoint AV
0x0008	Effective Occupied Heating Setpoint AV

Index	Application Usage
0x0009	Effective Unoccupied Heating Setpoint AV
0x000a	Effective Occupied Cooling Setpoint AV
0x000b	Effective Unoccupied Cooling Setpoint AV
0x000c	Effective Standby Heat Setpoint AV
0x000d	Effective Standby Cooling Setpoint AV
0x000e	Setpoint Offset AV
0x000f	Setpoint Shift AV
0x0200 to fffe	Vendor defined
0xffff	Other

5754 **3.14.11.19.3.2 Type = 0x01: Area in Square Metres**

5755 **Table 3-118. AV Types, Type = 0x01: Area in Square Metres**

Index	Application Usage
0x0000	Duct Area AV
0x0200- 0xFFFFE	Vendor defined
0xFFFFF	Other

5756 **3.14.11.19.3.3 Type = 0x02: Multiplier - Number**

5757 **Table 3-119. AV Types, Type = 0x02: Multiplier - Number**

Index	Application Usage
0x0000	Gain multiplier AV
0x0200 to 0xffff	Vendor defined
0xffff	Other

5758 **3.14.11.19.3.4 Type 0x03: Flow in Litres/Second**



5759

**Table 3-120. AV Types, Type = 0x03: Flow in Litres/Second**

Index	Application Usage
0x0000	Minimum Air Flow AV
0x0001	Maximum Air Flow AV
0x0002	Heating Minimum Air Flow AV
0x0003	Heating Maximum Air Flow AV
0x0004	Standby Minimum Air Flow AV
0x0005	Standby Maximum Air Flow AV
0x0200 to 0xffff	Vendor defined
0xffff	Other

5760 **3.14.11.19.4 Binary Inputs (BI) Types**

5761 Group = 0x03.

5762 The following sub-clauses describe the values when Type = 0x00 to 0x01. Types 0x02 to 0xFE are reserved, Type =  
 5763 0xFF indicates other.

5764 Present Value = 0 represents False, Off, Normal

5765 Present Value = 1 represents True, On, Alarm

5766 **3.14.11.19.4.1 Type = 0x00: Application Domain HVAC**

5767

**Table 3-121. BI Types, Type = 0x00: Application Domain HVAC**

Index	Application Usage
0x0000	2 Pipe Pump Status BI
0x0001	Air Proving Switch BI
0x0002	Alarm Reset BI
0x0003	Boiler Status BI
0x0004	Boiler Flow Status BI
0x0005	Boiler General Alarm BI
0x0006	Boiler High Temperature Alarm BI
0x0007	Boiler Isolation Valve Status BI

<b>Index</b>	<b>Application Usage</b>
0x0008	Boiler Maintenance Switch BI
0x0009	Boiler Pump Overload BI
0x000A	Boiler Pump Status BI
0x000B	Boiler Status BI
0x000C	Box Heating Alarm BI
0x000D	Chiller Alarm BI
0x000E	Chiller Chilled Water Flow Status BI
0x000F	Chiller Chilled Water Isolation Valve Status BI
0x0010	Chiller Condenser Water Flow Status BI
0x0011	Chiller Condenser Water Isolation Valve Status BI
0x0012	Chiller Maintenance Switch BI
0x0013	Chiller Status BI
0x0014	Chilled Water Expansion Tank Alarm BI
0x0015	Chilled Water Expansion Tank High Pressure Alarm BI
0x0016	Chilled Water Expansion Tank Low Pressure Alarm BI
0x0017	Chilled Water Expansion Tank Status BI
0x0018	Combustion Damper Status BI
0x0019	Cooling Alarm BI
0x001A	Cooling Pump Maintenance Switch BI
0x001B	Cooling Pump Overload BI
0x001C	Cooling Pump Status BI
0x001D	Condenser Water Expansion Tank Alarm BI
0x001E	Condenser Water Expansion Tank High Pressure Alarm BI
0x001F	Condenser Water Expansion Tank Low Pressure Alarm BI
0x0020	Condenser Water Expansion Tank Status BI

<b>Index</b>	<b>Application Usage</b>
0x0021	Condenser Water Pump Maintenance Switch BI
0x0022	Condenser Water Pump Overload BI
0x0023	Condenser Water Pump Status BI
0x0024	Decouple Loop Flow Direction BI
0x0025	Discharge Smoke BI
0x0026	Door Status BI
0x0027	Economizer Command BI
0x0028	Emergency Shutdown BI
0x0029	Equipment Tamper BI
0x002A	Energy Hold Off BI
0x002B	Exhaust Fan Maintenance Switch BI
0x002C	Exhaust Fan Overload BI
0x002D	Exhaust Fan Status BI
0x002E	Exhaust Filter Status BI
0x002F	Exhaust Smoke BI
0x0030	Expansion Tank Alarm BI
0x0031	Expansion Tank High Pressure Alarm BI
0x0032	Expansion Tank Low Pressure Alarm BI
0x0033	Expansion Tank Status BI
0x0034	Fan Control By Others BI
0x0035	Fan Overload BI
0x0036	Filter Monitoring BI
0x0037	Final Filter Status BI
0x0038	Free Cooling Availability BI
0x0039	Heat Recovery Pump Status BI

<b>Index</b>	<b>Application Usage</b>
0x003A	Heat Recovery Wheel Alarm BI
0x003B	Heat Recovery Wheel Maintenance Switch BI
0x003C	Heat Recovery Wheel Overload BI
0x003D	Heat Recovery Wheel Status BI
0x003E	Heating Alarm BI
0x003F	Heating/Cooling Pump Maintenance Switch BI
0x0040	Heating/Cooling Pump Overload BI
0x0041	High Humidity Limit BI
0x0042	High Static Pressure Fault BI
0x0043	High Temperature Limit Fault BI
0x0044	Humidifier Alarm BI
0x0045	Humidifier Maintenance Switch BI
0x0046	Humidifier Overload BI
0x0047	Humidifier Status BI
0x0048	Heat Exchanger Alarm BI
0x0049	Heat Exchanger Isolation Valve Status BI
0x004A	Heat Exchanger Maintenance Switch BI
0x004B	Lighting Status BI
0x004C	Low Static Pressure Fault BI
0x004D	Low Temperature Limit Fault BI
0x004E	Minimum Outdoor Air Damper End Switch BI
0x004F	Minimum Outdoor Air Fan Maintenance Switch BI
0x0050	Minimum Outdoor Air Fan Overload BI
0x0051	Minimum Outdoor Air Fan Status BI
0x0052	Minimum Outdoor Air Fan Variable Frequency Drive Fault BI

<b>Index</b>	<b>Application Usage</b>
0x0053	Occupancy BI
0x0054	Occupancy Sensor BI
0x0055	Primary Chilled Water Pump Maintenance Switch BI
0x0056	Primary Chilled Water Pump Overload BI
0x0057	Primary Chilled Water Pump Status BI
0x0058	Primary Chilled Water Pump Maintenance Switch BI
0x0059	Primary Chilled Water Pump Overload BI
0x005A	Primary Chilled Water Pump Status BI
0x005B	Pre-Filter Status BI
0x005C	Preheat Alarm BI
0x005D	Preheat Bonnet Switch BI
0x005E	Preheat Pump Maintenance Switch BI
0x005F	Preheat Pump Overload BI
0x0060	Preheat Pump Status BI
0x0061	Refrigerant Alarm BI
0x0062	Reheat Alarm BI
0x0063	Reheat Bonnet Switch BI
0x0064	Reheat Pump Maintenance Switch BI
0x0065	Reheat Pump Overload BI
0x0066	Reheat Pump Status BI
0x0067	Relief Fan Maintenance Switch BI
0x0068	Relief Fan Overload BI
0x0069	Relief Fan Status BI
0x006A	Relief Fan Variable Frequency Drive Fault BI
0x006B	Return Air Smoke BI

<b>Index</b>	<b>Application Usage</b>
0x006C	Return Fan Maintenance Switch BI
0x006D	Return Fan Overload BI
0x006E	Return Fan Status BI
0x006F	Return Fan VFD Fault BI
0x0070	Return Smoke BI
0x0071	Secondary Chilled Water Pump 1 Maintenance Switch BI
0x0072	Secondary Chilled Water Pump 1 Overload BI
0x0073	Secondary Chilled Water Pump 1 Status BI
0x0074	Secondary Chilled Water Pump 1 Maintenance Switch BI
0x0075	Secondary Chilled Water Pump 1 Overload BI
0x0076	Secondary Chilled Water Pump 1 Status BI
0x0077	Sideloop BI
0x0078	Generic Status BI
0x0079	Summer Winter BI
0x007A	Supplemental Heating Alarm BI
0x007B	Supplemental Heating Pump Maintenance Switch BI
0x007C	Supplemental Heating Pump Overload BI
0x007D	Supplemental Heating Pump Status BI
0x007E	Supply Fan Maintenance Switch BI
0x007F	Supply Fan Overload BI
0x0080	Supply Fan Status BI
0x0081	Supply Fan Variable Frequency Drive Fault BI
0x0082	Temporary Occupancy BI
0x0083	Tower Level Alarm BI
0x0084	Tower Level Status BI

Index	Application Usage
0x0085	Tower Temp BI
0x0086	Tower Vibration Alarm Status BI
0x0087	Tower Level Alarm BI
0x0088	Tower Level Switch BI
0x0089	Tower Temp Switch BI
0x008A	Tower Fan Isolation Valve Status BI
0x008B	Tower Fan Maintenance Switch BI
0x008C	Tower Fan Overload BI
0x008D	Tower Fan Status BI
0x008E	Unit Enable BI
0x008F	Unit Reset BI
0x0090	Window Status BI
0x0091	Zone Sensor Temporary Occupancy BI
0x0092	Air Proving Switch BI
0x0093	Primary Heating Status BI
0x0094	Primary Cooling Status BI
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5768 **3.14.11.19.4.2 Type = 0x01: Application Domain Security**

5769 **Table 3-122. BI Types, Type = 0x01: Application Domain Security**

Index	Application Usage
0x0000	Glass Breakage Detection
0x0001	Intrusion Detection
0x0002	Motion Detection
0x0003	Glass Breakage Detection

Index	Application Usage
0x0004	Zone Armed
0x0005	Glass Breakage Detection
0x0006	Smoke Detection
0x0007	Carbon Dioxide Detection
0x0008	Heat Detection
0x0200 to 0xFFFE	Vendor defined
0xFFFF	Other

5770 **3.14.11.19.5 Binary Output (BO) Types**

5771 Group = 0x04.

5772 The following sub-clauses describe the values when Type = 0x00 to 0x01. Types 0x02 to 0xFE are reserved, Type =  
5773 0xFF indicates other.

5774 Present Value = 0 represents False, Off, Normal

5775 Present Value = 1 represents True, On, Alarm

5776 **3.14.11.19.5.1 Type = 0x00: Application Domain HVAC**

5777 **Table 3-123. BO Types, Type = 0x00: Application Domain HVAC**

Index	Application Usage
0x0000	2 Pipe Circulation Pump BO
0x0001	2 Pipe Valve BO
0x0002	2 Pipe Valve Command BO
0x0003	Boiler BO
0x0004	Boiler Isolation Valve BO
0x0005	Boiler Pump BO
0x0006	Box Cooling 2 Position BO
0x0007	Box Heating 2 Position BO
0x0008	Box Heating Enable BO
0x0009	Box Heating Stage 1 BO



<b>Index</b>	<b>Application Usage</b>
0x000A	Box Heating Stage 2 BO
0x000B	Box Heating Stage 3 BO
0x000C	Chiller 1 Isolation Valve BO
0x000D	Chiller BO
0x000E	Chiller Chilled Water Isolation Valve BO
0x000F	Chiller Condenser Water Isolation Valve BO
0x0010	Combustion Damper BO
0x0011	Compressor Stage 1 BO
0x0012	Compressor Stage 2 BO
0x0013	Cooling Circulation Pump BO
0x0014	Cooling Stage 1 BO
0x0015	Cooling Stage 2 BO
0x0016	Cooling Stage 3 BO
0x0017	Cooling Stage 4 BO
0x0018	Cooling Stage 5 BO
0x0019	Cooling Stage 6 BO
0x001A	Cooling Stage 7 BO
0x001B	Cooling Stage 8 BO
0x001C	Cooling Valve BO
0x001D	Cooling Valve Command BO
0x001E	Chilled Water Pump BO
0x001F	Economizer Enable BO
0x0020	Exhaust Air Damper BO
0x0021	Exhaust Fan BO
0x0022	Fan BO

<b>Index</b>	<b>Application Usage</b>
0x0023	Fan Speed 1 BO
0x0024	Fan Speed 2 BO
0x0025	Fan Speed 3 BO
0x0026	Heat Recovery Pump BO
0x0027	Heat Recovery Valve BO
0x0028	Heat Recovery Wheel BO
0x0029	Heating Stage 1 BO
0x002A	Heating Stage 2 BO
0x002B	Heating Stage 3 BO
0x002C	Heating Valve BO
0x002D	Heating Valve Command BO
0x002E	Hot Gas Bypass Valve BO
0x002F	Humidification Stage 1 BO
0x0030	Humidification Stage 2 BO
0x0031	Humidification Stage 3 BO
0x0032	Humidification Stage 4 BO
0x0033	Humidifier Enable BO
0x0034	Heat Exchanger Isolation Valve BO
0x0035	Lighting BO
0x0036	Minimum Outside Air Damper BO
0x0037	Minimum Outside Air Fan BO
0x0038	Outside Air Damper BO
0x0039	Primary Chilled Water Pump 1 BO
0x003A	Plate-and-Frame Heat Exchanger Isolation Valve BO
0x003B	Primary Hot Water Pump BO

<b>Index</b>	<b>Application Usage</b>
0x003C	Primary Heat Exchange Pump BO
0x003D	Preheat Circulation Pump BO
0x003E	Preheat Enable BO
0x003F	Preheat Stage 1 BO
0x0040	Preheat Stage 2 BO
0x0041	Preheat Stage 3 BO
0x0042	Preheat Stage 4 BO
0x0043	Preheat Stage 5 BO
0x0044	Preheat Stage 6 BO
0x0045	Preheat Stage 7 BO
0x0046	Preheat Stage 8 BO
0x0047	Preheat Valve BO
0x0048	Reheat Circulation Pump BO
0x0049	Reheat Enable BO
0x004A	Reheat Stage 1 BO
0x004B	Reheat Stage 2 BO
0x004C	Reheat Stage 3 BO
0x004D	Reheat Stage 4 BO
0x004E	Reheat Stage 5 BO
0x004F	Reheat Stage 6 BO
0x0050	Reheat Stage 7 BO
0x0051	Reheat Stage 8 BO
0x0052	Relief Fan BO
0x0053	Return Fan BO
0x0054	Reversing Valve 1 BO

<b>Index</b>	<b>Application Usage</b>
0x0055	Reversing Valve 2 BO
0x0056	Secondary Chilled Water Pump BO
0x0057	Secondary Hot Water Pump BO
0x0058	Secondary Heat Exchange Pump BO
0x0059	Sideloop BO
0x005A	Sideloop Stage 1 BO
0x005B	Sideloop Stage 2 BO
0x005C	Sideloop Stage 3 BO
0x005D	Sideloop Stage 4 BO
0x005E	Sideloop Stage 5 BO
0x005F	Sideloop Stage 6 BO
0x0060	Sideloop Stage 7 BO
0x0061	Sideloop Stage 8 BO
0x0062	Steam Isolation Valve BO
0x0063	Supplemental Heating 2 Position BO
0x0064	Supplemental Heating Stage 1 BO
0x0065	Supplemental Heating Valve BO
0x0066	Supplemental Heating Enable BO
0x0067	Supplemental Heating Pump BO
0x0068	Supply Fan BO
0x0069	Tower Basin Heater BO
0x006A	Tower Basin Makeup BO
0x006B	Tower Basin Heater BO
0x006C	Tower Basin Makeup BO
0x006D	Tower Isolation Valve BO

Index	Application Usage
0x006E	Tower Fan BO
0x006F	Tower Fan Speed 1 BO
0x0070	Tower Fan Speed 2 BO
0x0071	Tower Fan Speed 3 BO
0x0072	Zone Heating Stage 1 BO
0x0073	Zone Heating Stage 2 BO
0x0074	Zone Heating Stage 3 BO
0x0075	Zone Heating Valve BO
0x0076	2 Pipe Circulation Pump BO
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5778 **3.14.11.19.5.2 Type = 0x02: Application Domain Security**

5779 **Table 3-124. BO Types, Type = 0x02: Application Domain Security**

Index	Application Usage
0x0000	Arm Disarm Command BO
0x0001	Occupancy Control BO
0x0002	Enable Control BO
0x0003	Access Control BO
0x0200 to 0xFFFFE	Vendor defined
0xFFFF	Other

5780 **3.14.11.19.6 Binary Value (BV) Types**

5781 Group = 0x05.

5782 The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF  
 5783 indicates other.

5784 Present Value = 0 represents False, Off, Normal

5785 Present Value = 1 represents True, On, Alarm

5786 **3.14.11.19.6.1 Type = 0x00**

5787

**Table 3-125. BV Types, Type = 0x00**

Index	Application Usage
0x0200- 0xFFFE	Vendor defined
0xFFFF	Other

5788 **3.14.11.19.7 Multistate Input (MI) Types**

5789 Group = 0x0D.

5790 The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF  
5791 indicates other.5792 **3.14.11.19.7.1 Type = 0x00: Application Domain HVAC**

5793

**Table 3-126. MI Types, Type = 0x00: Application Domain HVAC**

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emergency Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Supplemental Heat

Index	Application Usage [Number of States] States
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other

5794 **3.14.11.19.8 Multistate Output (MO) Types**

5795 Group = 0x0E.

5796 The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF  
 5797 indicates other.

5798 **3.14.11.19.8.1 Type = 0x00: Application Domain HVAC**

5799 **Table 3-127. MO Types, Type = 0x00: Application Domain HVAC**

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emerg Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Suppl Heat
0x0200- 0xFFFFE	Vendor defined

Index	Application Usage [Number of States] States
0xFFFF	Other

5800 **3.14.11.19.9 Multistate Value (MV) Types**

5801 Group = 0x13.

5802 The following sub-clauses describe the values when Type = 0x00. Types 0x01 to 0xFE are reserved, Type = 0xFF  
5803 indicates other.5804 **3.14.11.19.9.1 Type = 0x00: Application Domain HVAC**5805 **Table 3-128. MV Types, Type = 0x00: Application Domain HVAC**

Index	Application Usage [Number of States] States
0x0000	[3] Off, On, Auto
0x0001	[4] Off, Low, Medium, High
0x0002	[7] Auto, Heat, Cool, Off, Emerg Heat, Fan Only, Max Heat
0x0003	[4] Occupied, Unoccupied, Standby, Bypass
0x0004	[3] Inactive, Active, Hold
0x0005	[8] Auto, Warm-up, Water Flush, Autocalibration, Shutdown Open, Shutdown Closed, Low Limit, Test and Balance
0x0006	[6] Off, Auto, Heat Cool, Heat Only, Cool Only, Fan Only
0x0007	[3] High, Normal, Low
0x0008	[4] Occupied, Unoccupied, Startup, Shutdown
0x0009	[3] Night, Day, Hold
0x000A	[5] Off, Cool, Heat, Auto, Emergency Heat
0x000B	[7] Shutdown Closed, Shutdown Open, Satisfied, Mixing, Cooling, Heating, Suppl Heat
0x0200- 0xFFFFE	Vendor defined
0xFFFF	Other



5806  
 5807 All other group values are currently reserved

## 5808 3.15 Diagnostics

---

### 5809 3.15.1 Overview

5810 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 5811 etc.

5812 The diagnostics cluster provides access to information regarding the operation of the stack over time. This information  
 5813 is useful to installers and other network administrators who wish to know how a particular device is functioning on  
 5814 the network.

5815 The Diagnostics Cluster needs to understand the performance of the network over time in order to isolate network  
 5816 routing issues.

5817 While it is not absolutely essential, it is recommended that server attributes be stored in persistent memory. This  
 5818 especially makes sense if for instance some stack behavior were causing a device to reset. Without storing the  
 5819 associated server attributes in persistent memory there would be no way to analyze what was causing the reset  
 5820 behavior.

#### 5821 3.15.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added
2	CCB 2309 2212 2333

#### 5822 3.15.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	DIAG

#### 5823 3.15.1.3 Cluster Identifiers

Identifier	Name
0x0b05	Diagnostics

### 5824 3.15.2 Server

#### 5825 3.15.2.1 Attributes

5826 The server attributes in the diagnostics cluster are broken up into several attribute sets listed in Table 3-129.

5827

**Table 3-129. Server Attribute Sets of the Diagnostics Cluster**

Attribute Set Identifier	Description
0x0000	Hardware Information
0x0100	Stack/Network Information

5828 **3.15.2.1.1 Hardware Information Attribute Set**

5829

**Table 3-130. Hardware Information Attribute Set**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>NumberOfResets</i>	uint16	0x0000 to 0xffff	R	0x00000000	O
0x0001	<i>PersistentMemoryWrites</i>	uint16	0x0000 to 0xffff	R	0x00000000	O

5830 **3.15.2.1.1.1 NumberOfResets Attribute**

5831 An attribute that is incremented each time the device resets. A reset is defined as any time the device restarts. This is  
5832 not the same as a reset to factory defaults, which SHOULD clear this and all values.

5833 **3.15.2.1.1.2 PersistentMemoryWrites Attribute**

5834 This attribute keeps track of the number of writes to persistent memory. Each time that the device stores a token in  
5835 persistent memory it will increment this value.

5836 **3.15.2.1.2 Stack / Network Information Attribute Set**

5837 Note that many of the counters in this attribute set (Table 3-131) will wrap quickly. They SHOULD be read frequently  
5838 during periods of network interrogation in order to avoid missing points where the counters roll over.

5839

**Table 3-131. Stack / Network Information Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
0x0100	<i>MacRxBcast</i>	uint32	0x00000000 to 0xffffffff <sup>38</sup>	R	0	O
0x0101	<i>MacTxBcast</i>	uint32	0x00000000 to 0xffffffff	R	0	O
0x0102	<i>MacRxUcast</i>	uint32	0x00000000 to 0xffffffff	R	0	O
0x0103	<i>MacTxUcast</i>	uint32	0x00000000 to 0xffffffff	R	0	O
0x0104	<i>MacTxUcastRetry</i>	uint16	0x0000 to 0xffff	R	0	O
0x0105	<i>MacTxUcastFail</i>	uint16	0x0000 to 0xffff	R	0	O
0x0106	<i>APSRxBcast</i>	uint16	0x0000 to 0xffff	R	0	O
0x0107	<i>APSTxBcast</i>	uint16	0x0000 to 0xffff	R	0	O

<sup>38</sup> CCB 2309

0x0108	<i>APSRxUcast</i>	uint16	0x0000 to 0xffff	R	0	O
0x0109	<i>APSTxUcastSuccess</i>	uint16	0x0000 to 0xffff	R	0	O
0x010A	<i>APSTxUcastRetry</i>	uint16	0x0000 to 0xffff	R	0	O
0x010B	<i>APSTxUcastFail</i>	uint16	0x0000 to 0xffff	R	0	O
0x010C	<i>RouteDiscInitiated</i>	uint16	0x0000 to 0xffff	R	0	O
0x010D	<i>NeighborAdded</i>	uint16	0x0000 to 0xffff	R	0	O
0x010E	<i>NeighborRemoved</i>	uint16	0x0000 to 0xffff	R	0	O
0x010F	<i>NeighborStale</i>	uint16	0x0000 to 0xffff	R	0	O
0x0110	<i>JoinIndication</i>	uint16	0x0000 to 0xffff	R	0	O
0x0111	<i>ChildMoved</i>	uint16	0x0000 to 0xffff	R	0	O
0x0112	<i>NWKFCFailure</i>	uint16	0x0000 to 0xffff	R	0	O
0x0113	<i>APSFCEFailure</i>	uint16	0x0000 to 0xffff	R	0	O
0x0114	<i>APSUnauthorizedKey</i>	uint16	0x0000 to 0xffff	R	0	O
0x0115	<i>NWKDecryptFailures</i>	uint16	0x0000 to 0xffff	R	0	O
0x0116	<i>APSDDecryptFailures</i>	uint16	0x0000 to 0xffff	R	0	O
0x0117	<i>PacketBufferAllocateFailures</i>	uint16	0x0000 to 0xffff	R	0	O
0x0118	<i>RelayedUcast</i>	uint16	0x0000 to 0xffff	R	0	O
0x0119	<i>PhytoMACQueueLimitReached</i>	uint16	0x0000 to 0xffff	R	0	O
0x011A	<i>PacketValidatedDropCount</i>	uint16	0x0000 to 0xffff	R	0	O
0x011B	<i>AverageMACRetryPerAPSMes- sageSent</i>	uint16	0x0000 to 0xffff	R	0	O
0x011C	<i>LastMessageLQI</i>	uint8	0x00 to 0xff	R	0	O
0x011D	<i>LastMessageRSSI</i>	int8 <sup>39</sup>	-127 to 127	R	0	O

5840 **3.15.2.1.2.1 MacRxBcast Attribute**

5841 A counter that is incremented each time the MAC layer receives a broadcast.

5842 **3.15.2.1.2.2 MacTxBcast Attribute**

<sup>39</sup> CCB 2212

- 5843 A counter that is incremented each time the MAC layer transmits a broadcast.
- 5844 **3.15.2.1.2.3 *MacRxUcast Attribute***
- 5845 A counter that is incremented each time the MAC layer receives a unicast.
- 5846 **3.15.2.1.2.4 *MacTxUcast Attribute***
- 5847 A counter that is incremented each time the MAC layer transmits a unicast.
- 5848 **3.15.2.1.2.5 *MacTxUcastRetry Attribute***
- 5849 A counter that is incremented each time the MAC layer retries a unicast.
- 5850 **3.15.2.1.2.6 *MacTxUcastFail Attribute***
- 5851 A counter that is incremented each time the MAC layer fails to send a unicast.
- 5852 **3.15.2.1.2.7 *APSRxBcast Attribute***
- 5853 A counter that is incremented each time the APS layer receives a broadcast.
- 5854 **3.15.2.1.2.8 *APSTxBcast Attribute***
- 5855 A counter that is incremented each time the APS layer transmits a broadcast.
- 5856 **3.15.2.1.2.9 *APSRxUcast Attribute***
- 5857 A counter that is incremented each time the APS layer receives a unicast.
- 5858 **3.15.2.1.2.10 *APSTxUcastSuccess Attribute***
- 5859 A counter that is incremented each time the APS layer successfully transmits a unicast.
- 5860 **3.15.2.1.2.11 *APSTxUcastRetry Attribute***
- 5861 A counter that is incremented each time the APS layer retries the sending of a unicast.
- 5862 **3.15.2.1.2.12 *APSTxUcastFail Attribute***
- 5863 A counter that is incremented each time the APS layer fails to send a unicast.
- 5864 **3.15.2.1.2.13 *RouteDisclnitiated Attribute***
- 5865 A counter that is incremented each time a route request is initiated<sup>40</sup>.
- 5866 **3.15.2.1.2.14 *NeighborAdded Attribute***
- 5867 A counter that is incremented each time an entry is added to the neighbor table.
- 5868 **3.15.2.1.2.15 *NeighborRemoved Attribute***
- 5869 A counter that is incremented each time an entry is removed from the neighbor table.
- 5870 **3.15.2.1.2.16 *NeighborStale Attribute***
- 5871 A counter that is incremented each time a neighbor table entry becomes stale because the neighbor has not been heard from.
- 5872
- 5873 **3.15.2.1.2.17 *JoinIndication Attribute***

---

<sup>40</sup> CCB 2333

- 5874 A counter that is incremented each time a node joins or rejoins the network via this node.
- 5875 **3.15.2.1.2.18 *ChildMoved Attribute***
- 5876 A counter that is incremented each time an entry is removed from the child table.
- 5877 **3.15.2.1.2.19 *NWKFCFailure Attribute***
- 5878 A counter that is incremented each time a message is dropped at the network layer because the APS frame counter  
5879 was not higher than the last message seen from that source.
- 5880 **3.15.2.1.2.20 *APSFCEFailure Attribute***
- 5881 A counter that is incremented each time a message is dropped at the APS layer because the APS frame counter was  
5882 not higher than the last message seen from that source.
- 5883 **3.15.2.1.2.21 *APSUnauthorizedKey Attribute***
- 5884 A counter that is incremented each time a message is dropped at the APS layer because it had APS encryption but the  
5885 key associated with the sender has not been authenticated, and thus the key is not authorized for use in APS data  
5886 messages.
- 5887 **3.15.2.1.2.22 *NWKDecryptFailures Attribute***
- 5888 A counter that is incremented each time a NWK encrypted message was received but dropped because decryption  
5889 failed.
- 5890 **3.15.2.1.2.23 *APSDecryptFailures Attribute***
- 5891 A counter that is incremented each time an APS encrypted message was received but dropped because decryption  
5892 failed.
- 5893 **3.15.2.1.2.24 *PacketBufferAllocateFailures Attribute***
- 5894 A counter that is incremented each time the stack failed to allocate a packet buffers. This doesn't necessarily mean  
5895 that the packet buffer count was 0 at the time, but that the number requested was greater than the number free.
- 5896 **3.15.2.1.2.25 *RelayedUcast Attribute***
- 5897 A counter that is incremented each time a unicast packet is relayed.
- 5898 **3.15.2.1.2.26 *PacketValidateDropCount Attribute***
- 5899 A counter that is incremented each time a packet was dropped due to a packet validation error. This could be due to  
5900 length or other formatting problems in the packet.
- 5901 **3.15.2.1.2.27 *AverageMACRetryPerAPSMessagesent Attribute***
- 5902 A counter that is equal to the average number of MAC retries needed to send an APS message.
- 5903 **3.15.2.1.2.28 *LastMessageLQI Attribute***
- 5904 This is the Link Quality Indicator for the last message received. There is no current agreed upon standard for  
5905 calculating the LQI. For some implementations LQI is related directly to RSSI for others it is a function of the number  
5906 of errors received over a fixed number of bytes in a given message. The one thing that has been agreed is that the Link  
5907 Quality Indicator is a value between 0 and 255 where 0 indicates the worst possible link and 255 indicates the best  
5908 possible link. Note that for a device reading the Last Message LQI the returned value SHALL be the LQI for the read  
5909 attribute message used to read the attribute itself.
- 5910 **3.15.2.1.2.29 *LastMessageRSSI Attribute***

5911 This is the receive signal strength indication for the last message received. As with Last Message LQI, a device reading  
5912 the Last Message RSSI, the returned value SHALL be the RSSI of the read attribute message used to read the attribute  
5913 itself.

### 5914 **3.15.2.2 Commands**

5915 There are no commands received by the server side of the diagnostics cluster.

### 5916 **3.15.3 Client**

5917 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
5918 specific commands.

## 5919 **3.16 Poll Control**

---

### 5920 **3.16.1 Overview**

5921 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
5922 etc.

5923 This cluster provides a mechanism for the management of an end device's MAC Data Request rate. For the purposes  
5924 of this cluster, the term "poll" always refers to the sending of a MAC Data Request from the end device to the end  
5925 device's parent.

5926 This cluster can be used for instance by a configuration device to make an end device responsive for a certain period  
5927 of time so that the device can be managed by the controller.

5928 This cluster is composed of a client and server. The end device implements the server side of this cluster. The server  
5929 side contains several attributes related to the MAC Data Request rate for the device. The client side implements  
5930 commands used to manage the poll rate for the device.

5931 The end device which implements the server side of this cluster sends a query to the client on a predetermined interval  
5932 to see if the client would like to manage the poll period of the end device in question. When the client side of the  
5933 cluster hears from the server it has the opportunity to respond with configuration data to either put the end device in a  
5934 short poll mode or let the end device continue to function normally.

#### 5935 **3.16.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added; CCB 1815 1822 1833
2	CCB 2319 2329

#### 5936 **3.16.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	POLL

5937 **3.16.1.3 Cluster Identifiers**

Identifier	Name
0x0020	Poll Control

5938 **3.16.2 Terminology**

5939 MAC Data Request Rate: The MAC Data Request rate or simply “poll rate” is the frequency with which an end device  
 5940 sends a MAC Data Request to its parent. A parent device is only required to store a single message for its child for  
 5941 7.68 seconds. Therefore if an end device wants to retrieve messages from its parent, it must send a MAC Data Request  
 5942 every 7.68 seconds.

5943 Generally, end devices have two different rates at which they send MAC Data Polls to their parents. A slower rate for  
 5944 when the device is not expecting data (Long Poll Interval) and a faster rate (Short Poll Interval) for when the device  
 5945 is expecting data.

5946 End devices only know that they are expecting data when they have initiated some sort of transaction. This cluster  
 5947 provides a mechanism for forcing this state to make the end device responsive to asynchronous messaging.

5948 Long Poll Interval: The amount of time between MAC Data Requests when the device is in its normal operating state  
 5949 and not expecting any messages.

5950 Short Poll Interval: The amount of time between MAC Data Requests when the device is either expecting data or has  
 5951 been put into “Fast Poll Mode” by the controlling device.

5952 Fast Poll Mode: When the device is polling frequently to retrieve data from its parent we say that the device is in “Fast  
 5953 Poll Mode”. The entire purpose of this cluster is to provide a means of managing when an end device goes into and  
 5954 out of Fast Poll Mode so that it can be made responsive for a controlling device.

5955 **3.16.3 Commissioning Process**

5956 Poll Control Cluster Clients SHALL configure bindings on the device implementing the Poll Control Cluster Server  
 5957 so that they will receive the regular check-in command on the configured *Check-In Interval*. This can be done during  
 5958 the configuration period on the end device implementing the Poll Control Cluster Server during which it is in fast poll  
 5959 mode. The device that implements the Poll Control Cluster Server SHALL check its bindings on the configured check-  
 5960 in Interval. If it has any bindings related to any endpoint and the Poll Control Cluster, it will send a check-in command  
 5961 out on that binding.

5962 **3.16.4 Server**

5963 **3.16.4.1 Attributes**

5964 The server side of this cluster contains certain attributes (Table 3-132) associated with the poll period.  
 5965 *CheckInIntervalMin*, *LongPollIntervalMin*, *FastPollTimeoutMaximum* attributes are optional; however, if they are  
 5966 not supported, you could end up with a lot of chatter on the network as clients and servers attempt to negotiate the poll  
 5967 period. It is therefore recommended that these attributes be supported.

5968 **Table 3-132. Server Attributes**

Identifier	Name	Type	Range	Acc	Default	M/O
0x0000	<i>Check-inInterval</i>	uint32	0x0 to 0x6E0000	RW	0x3840 (1 hr.)	M
0x0001	<i>LongPoll Interval</i>	uint32	0x04 to 0x6E0000	R	0x14 (5 sec)	M
0x0002	<i>ShortPollInterval</i>	uint16	0x01 to 0xffff	R	0x02 (2 qs)	M

Identifier	Name	Type	Range	Acc	Default	M/O
0x0003	<i>FastPollTimeout</i>	uint16	0x01 to 0xffff	RW	0x28 (10 sec.)	M
0x0004	<i>Check-inIntervalMin</i>	uint32	-	R	0	O
0x0005	<i>LongPollIntervalMin</i>	uint32	-	R	0	O
0x0006	<i>FastPollTimeoutMax</i>	uint16	-	R	0	O

#### 5969 **3.16.4.1.1** *Check-inInterval* Attribute

5970 The Poll Control server is responsible for checking in with the poll control client periodically to see if the poll control  
5971 client wants to modify the poll rate of the poll control server. This is due to the fact that the Poll Control server is  
5972 implemented on an end device that MAY have an unpredictable sleep-wake cycle.

5973 The *Check-inInterval* represents the default amount of time between check-ins by the poll control server with the poll  
5974 control client. The *Check-inInterval* is measured in quarterseconds. A value of 0 indicates that the Poll Control Server  
5975 is turned off and the poll control server will not check-in with the poll control client.

5976 The Poll Control Server checks in with the Poll Control Client by sending a Check-in command to the Client. This  
5977 value SHOULD be longer than the *LongPollInterval* attribute. If the Client writes an invalid attribute value (Example:  
5978 Out of Range as defined in Table 3-132 or a value smaller than the optional *Check-inIntervalMin* attribute value or a  
5979 value smaller than the *LongPollInterval* attribute value), the Server SHOULD return Write Attributes Response with  
5980 an error status not equal to SUCCESS(0x00).

5981 The Poll Control Client will hold onto the actions or messages for the Poll Control Server at the application level until  
5982 the Poll Control Server checks in with the Poll Control Client.

#### 5983 **3.16.4.1.2** *LongPollInterval* Attribute

5984 An end device that implements the Poll Control server MAY optionally expose a *LongPollInterval* attribute. The Long  
5985 Poll Interval represents the maximum amount of time in quarterseconds between MAC Data Requests from the end  
5986 device to its parent.

5987 The *LongPollInterval* defines the frequency of polling that an end device does when it is NOT in fast poll mode. The  
5988 *LongPollInterval* SHOULD be longer than the *ShortPollInterval* attribute but shorter than the *Check-inInterval*  
5989 attribute.

5990 A value of 0xffffffff is reserved to indicate that the device does not have or does not know its long poll interval.

#### 5991 **3.16.4.1.3** *ShortPollInterval* Attribute

5992 An end device that implements the Poll Control server MAY optionally expose the *ShortPollInterval* attribute. The  
5993 *ShortPollInterval* represents the number of quarterseconds that an end device waits between MAC Data Requests to  
5994 its parent when it is expecting data (i.e., in fast poll mode).

#### 5995 **3.16.4.1.4** *FastPollTimeout* Attribute

5996 The *FastPollTimeout* attribute represents the number of quarterseconds that an end device will stay in fast poll mode  
5997 by default. It is suggested that the *FastPollTimeout* attribute value be greater than 7.68 seconds.



5998 The Poll Control Cluster Client MAY override this value by indicating a different value in the Fast Poll Duration  
5999 argument in the Check-in Response command. If the Client writes a value out of range as defined in Table 3-132 or  
6000 greater than the optional *FastPollTimeoutMax* attribute value if supported, the Server SHOULD return a Write  
6001 Attributes Response with a status of INVALID\_VALUE. An end device that implements the Poll Control server can  
6002 be put into a fast poll mode during which it will send MAC Data Requests to its parent at the frequency of its  
6003 configured *ShortPollInterval* attribute. During this period of time, fast polling is considered active. When the device  
6004 goes into fast poll mode, it is required to send MAC Data Requests to its parent at an accelerated rate and is thus more  
6005 responsive on the network and can receive data asynchronously from the device implementing the Poll Control Cluster  
6006 Client.

#### 6007 **3.16.4.1.5 Check-inIntervalMin Attribute**

6008 The Poll Control Server MAY optionally provide its own minimum value for the *Check-inInterval* to protect against  
6009 the *Check-inInterval* being set too low and draining the battery on the end device implementing the Poll Control  
6010 Server.

#### 6011 **3.16.4.1.6 LongPollIntervalMin Attribute**

6012 The Poll Control Server MAY optionally provide its own minimum value for the *LongPollInterval* to protect against  
6013 another device setting the value to too short a time resulting in an inadvertent power drain on the device.

#### 6014 **3.16.4.1.7 FastPollTimeoutMax Attribute**

6015 The Poll Control Server MAY optionally provide its own maximum value for the *FastPollTimeout* to avoid it being  
6016 set to too high a value resulting in an inadvertent power drain on the device.

### 6017 **3.16.4.2 Attribute Settings and Battery Life Considerations**

6018 The Poll Control Cluster is used on end devices that MAY be battery powered. In order to conserve battery life, it is  
6019 important that the Poll Control Server maintain certain boundaries for the setting of the *Check-inInterval*,  
6020 *LongPollInterval* and the *ShortPollInterval*. Therefore, while these attributes are all Readable and Writeable, it is  
6021 possible that a battery-powered device might maintain its own boundary for the min and max of each of these  
6022 attributes. The end device implementing the Poll Control Cluster Server MAY define its own boundaries for these  
6023 attributes in order to protect itself against a power drain due to improper configuration.

6024 For instance, a battery powered device MAY not allow another device to set its *Check-inInterval* to too short a value  
6025 or its *FastPollTimeout* to too long an interval because it might cause the device to send too frequent check-in messages  
6026 on the network and stay in fast poll mode for too long a time resulting in a drain on the battery.

6027 The Check-inInterval, LongPollInterval and ShortPollInterval SHOULD be set such that:

6028 **Check-in Interval >= Long Poll Interval >= Short Poll Interval**

6029 The default values chosen for this cluster are:

6030 **Check-in Interval = 1 hour = 0x3840 quarterseconds**

6031 **Long Poll Interval = 5 seconds = 0x14 quarterseconds**

6032 **Short Poll Interval = 2 quarterseconds = 0x02 quarterseconds**

6033 **Fast Poll Timeout = 10 seconds = 0x28 quarterseconds**

6034 Note that for the Check-in Interval, 0 is a special value and does not apply to this equation.

### 6035 3.16.4.3 Commands

6036 Table 3-133. Commands Generated by the Poll Control Server

Command ID	Description	Mandatory/Optional
0x00	Check-in	M

### 6037 3.16.4.4 Check-in Command

6038 The Poll Control Cluster server sends out a Check-in command to the devices to which it is paired based on the server's  
6039 *Check-inInterval* attribute. It does this to find out if any of the Poll Control Cluster Clients with which it is paired are  
6040 interested in having it enter fast poll mode so that it can be managed. This request is sent out based on either the  
6041 *Check-inInterval*, or the next Check-in value in the Fast Poll Stop Request generated by the Poll Control Cluster  
6042 Client.

6043 The Check-in command expects a Check-in Response command to be sent back from the Poll Control Client. If the  
6044 Poll Control Server does not receive a Check-in response back from the Poll Control Client up to 7.68 seconds it is  
6045 free to return to polling according to the *LongPollInterval*.

#### 6046 3.16.4.4.1 Payload Format

6047 There is no payload for this command.

#### 6048 3.16.4.4.2 Effect on Receipt

6049 Upon receipt of the Check-in command, the Poll Control Cluster client will respond with a Check-in Response  
6050 command indicating that the server SHOULD or SHOULD not begin fast poll mode.

## 6051 3.16.5 Client

### 6052 3.16.5.1 Attributes

6053 There are no attributes on the client side of the Poll Control Cluster.

### 6054 3.16.5.2 Commands

6055 Table 3-134. Commands Generated by the Poll Control Client

Command ID	Description	Mandatory/Optional
0x00	Check-in Response	M
0x01	Fast Poll Stop	M
0x02	Set Long Poll Interval	O
0x03	Set Short Poll Interval	O

6056

### 6057 3.16.5.3 Check-in Response Command

6058 The Check-in Response is sent in response to the receipt of a Check-in command. The Check-in Response is used by  
 6059 the Poll Control Client to indicate whether it would like the device implementing the Poll Control Cluster Server to  
 6060 go into a fast poll mode and for how long. If the Poll Control Cluster Client indicates that it would like the device to  
 6061 go into a fast poll mode, it is responsible for telling the device to stop fast polling when it is done sending messages  
 6062 to the fast polling device.

6063 If the Poll Control Server receives a Check-In Response from a client for which there is no binding (unbound), it  
 6064 SHOULD respond with a Default Response with a status value indicating ACTION\_DENIED.

6065 If the Poll Control Server receives a Check-In Response from a client for which there is a binding (bound) with an  
 6066 invalid fast poll timeout<sup>41</sup>, it SHOULD respond with a Default Response with status INVALID\_VALUE.

6067 If the Poll Control Server receives a Check-In Response from a bound client after temporary fast poll mode is  
 6068 completed it SHOULD respond with a Default Response with a status value indicating TIMEOUT.

6069 In all of the above cases, the Server SHALL respond with a Default Response not equal to SUCCESS<sup>42</sup>.

#### 6070 3.16.5.3.1 Payload Format

6071 **Figure 3-63. Format of the Check-in Response Payload**

<b>Octets</b>	1	2
<b>Data Type</b>	bool	uint16
<b>Field Name</b>	Start Fast Polling	Fast Poll Timeout

##### 6072 3.16.5.3.1.1 Start Fast Polling

6073 This Boolean value indicates whether or not the Poll Control Server device SHOULD begin fast polling or not. If the  
 6074 Start Fast Polling value is true, the server device is EXPECTED to begin fast polling until the Fast Poll Timeout has  
 6075 expired. If the Start Fast Polling argument is false, the Poll Control Server MAY continue in normal operation and is  
 6076 not required to go into fast poll mode.

##### 6077 3.16.5.3.1.2 Fast Poll Timeout

6078 The Fast Poll Timeout value indicates the number of quarterseconds during which the device SHOULD continue fast  
 6079 polling. If the Fast Poll Timeout value is 0, the device is EXPECTED to continue fast polling until the amount of time  
 6080 indicated in the *FastPollTimeout* attribute has elapsed or it receives a Fast Poll Stop command. If the Start Fast Polling  
 6081 argument is false, the Poll Control Server MAY ignore the Fast Poll Timeout argument.

6082 The Fast Poll Timeout argument temporarily overrides the *FastPollTimeout* attribute on the Poll Control Cluster  
 6083 Server for the fast poll mode induced by the Check-in Response command. This value is not EXPECTED to overwrite  
 6084 the stored value in the *FastPollTimeout* attribute.

6085 If the *FastPollTimeout* parameter in the CheckInResponse command is greater than the *FastPollTimeoutMax* attribute  
 6086 value, the Server Device SHALL respond with a default response of error status not equal to SUCCESS. It is suggested  
 6087 to use the Error Status of ZCL\_INVALID\_FIELD (0x85).

<sup>41</sup> CCB 2330

<sup>42</sup> CCB 2319 no such status code as ZCL\_SUCCESS

### 6088 3.16.5.4 Fast Poll Stop Command

6089 The Fast Poll Stop command is used to stop the fast poll mode initiated by the Check-in response. The Fast Poll Stop  
6090 command has no payload.

6091 If the Poll Control Server receives a Fast Poll Stop from an unbound client it SHOULD send back a DefaultResponse  
6092 with a value field indicating ACTION\_DENIED”. The Server SHALL respond with a DefaultResponse not equal to  
6093 SUCCESS.

6094 If the Poll Control Server receives a Fast Poll Stop command from a bound client but it is unable to stop fast polling  
6095 due to the fact that there is another bound client which has requested that polling continue it SHOULD respond with  
6096 a Default Response with a status of “ACTION\_DENIED”

6097 If a Poll Control Server receives a Fast Poll Stop command from a bound client but it is not FastPolling it SHOULD  
6098 respond with a Default Response with a status of ACTION\_DENIED.

### 6099 3.16.5.5 Set Long Poll Interval Command

6100 The Set Long Poll Interval command is used to set the Read Only *LongPollInterval* attribute.

6101 When the Poll Control Server receives the Set Long Poll Interval Command, it SHOULD check its internal minimal  
6102 limit and the attributes relationship defined in 3.16.4.2 if the new Long Poll Interval is acceptable. If the new value is  
6103 acceptable, the new value SHALL be saved to the *LongPollInterval* attribute. If the new value is not acceptable, the  
6104 Poll Control Server SHALL send a default response of INVALID\_VALUE (0x87) and the *LongPollInterval* attribute  
6105 value is not updated.

#### 6106 3.16.5.5.1 Payload Format

6107 **Figure 3-64. Format of the Set Long Poll Interval Command Payload**

<b>Octets</b>	4
<b>Data Type</b>	uint32
<b>Field Name</b>	NewLongPollInterval

### 6108 3.16.5.6 Set Short Poll Interval Command

6109 The Set Short Poll Interval command is used to set the Read Only *ShortPollInterval* attribute.

6110 When the Poll Control Server receives the Set Short Poll Interval Command, it SHOULD check its internal minimal  
6111 limit and the attributes relationship defined in 3.16.4.2 if the new Short Poll Interval is acceptable. If the new value is  
6112 acceptable, the new value SHALL be saved to the *ShortPollInterval* attribute. If the new value is not acceptable, the  
6113 Poll Control Server SHALL send a default response of INVALID\_VALUE (0x87) and the *ShortPollInterval* attribute  
6114 value is not updated.

6115 **3.16.5.6.1 Payload Format**

6116 **Figure 3-65. Format of the Set Short Poll Interval Command Payload**

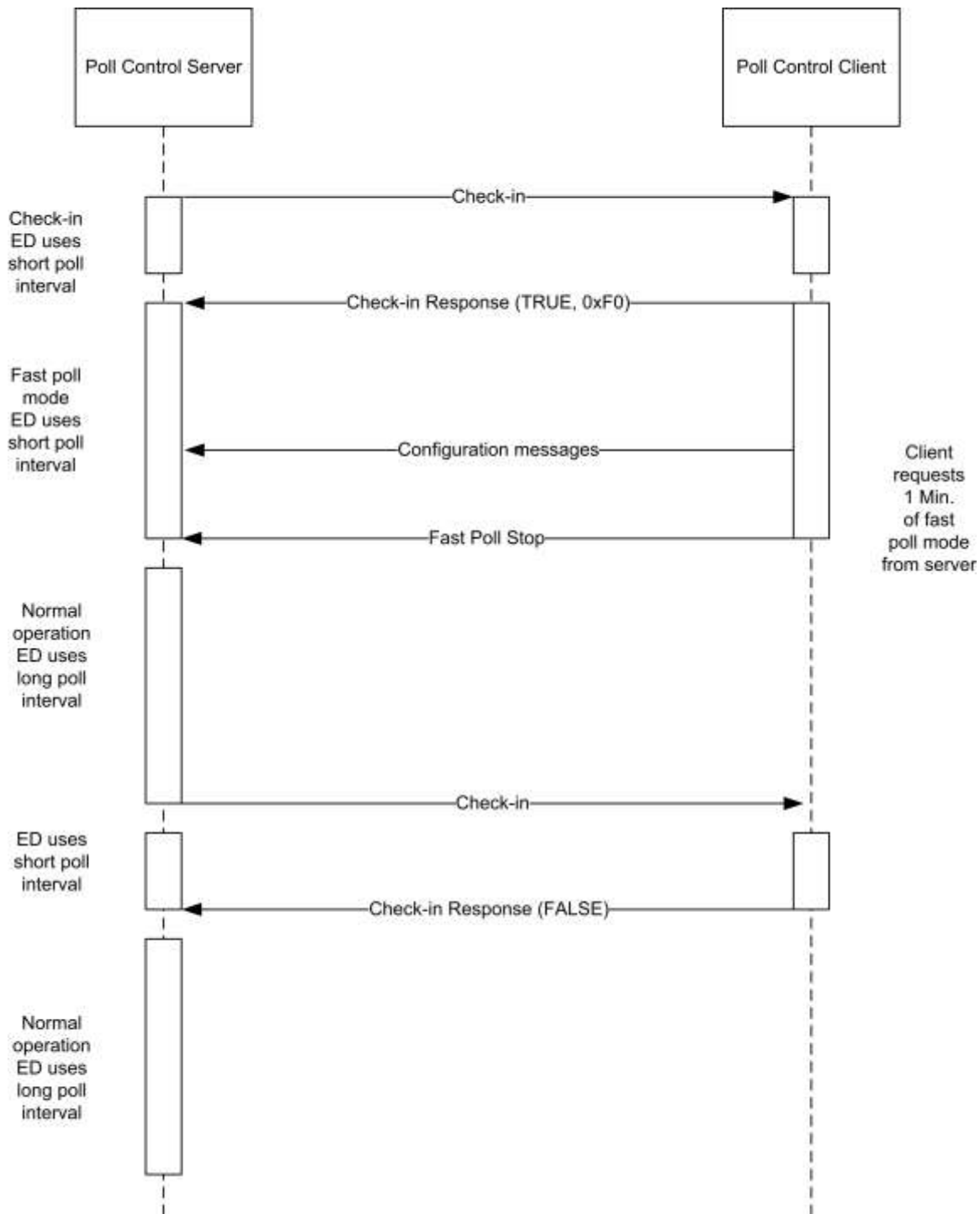
<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	New Short Poll Interval

6117 **3.16.6 Poll Control Cluster Sequence Diagram**

6118 What follows is a typical sequence interaction between the client and server sides of the Poll Control Cluster.

6119

Figure 3-66. Poll Control Cluster Sequence Diagram



6120

### 6121 3.16.6.1 Guaranteed Consistent Check-In Interval

6122 Provided that the *Check-inInterval* attribute value stays constant, the interval between two Check In commands is  
 6123 guaranteed. The *Check-inInterval* SHOULD be kept independent regardless of when the Check-In Response or Fast  
 6124 Poll Stop command is received.

### 6125 **3.16.6.2 Multiple Poll Control Client**

6126 When the *Check-inInterval* expires, the Server SHOULD send parallel Check-In commands to all paired client  
6127 devices.

6128 The server SHOULD then enter a temporary Fast Poll Mode, with a fixed manufacturer-specific predefined check in  
6129 timeout duration (t1), to wait for the Check-In Response Messages from all paired device.

6130 Once the server received all the Check-In Response or if the temporary Fast Poll Mode timeout (t1), the server  
6131 SHOULD then gather the information from all Check-In Response messages and determine the longest Fast Poll  
6132 Timeout (t2) duration.

6133 The Server device SHALL stay in the Fast Poll Mode for the longest Fast Poll Timeout (t2) duration. The server  
6134 device MAY end fast poll mode before the longest fast poll timeout if it is able to determine that every start request  
6135 from the paired device has been stopped explicitly by the Fast Poll Stop command or implicitly by a timeout.

6136 For example:

6137 Device A implements a poll control server, devices B and C implement poll control clients. Device A sends a check-  
6138 in command to both B and C. Both B and C respond with check-in response command requesting a fast poll start.  
6139 Assume B requests fast polling for 5 minutes and C requests fast polling for 10 minutes. If C sends a fast poll stop  
6140 command after 7 minutes, device A MAY immediately end fast polling upon receipt of this command since the fast  
6141 poll period requested by B would have expired after only 5 minutes (before the command from C was received).

### 6142 **3.16.6.3 Check-in Interval Attribute Changed**

6143 When the *Check-inInterval* attribute is changed (provided that the new value is valid and within acceptable range),  
6144 the device SHOULD reset the internal check-in interval timer and send a check-in command according to the  
6145 new *Check-inInterval* value.

## 6146 **3.17 Power Profile**

---

6147 This section describes the Power Profile cluster.

### 6148 **3.17.1 Overview**

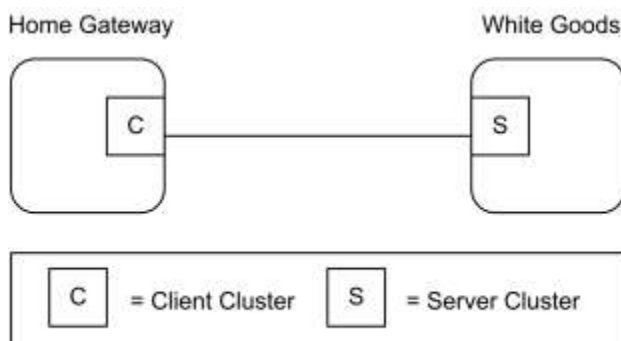
6149 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
6150 etc.

6151 This cluster provides an interface for transferring power profile information from a device (e.g., White Goods) to a  
6152 controller (e.g., the Home Gateway). The Power Profile can be solicited by client side (request command) or can be  
6153 notified directly from the device (server side). The Power Profile represents a forecast of the energy that a device is  
6154 able to predict. It is split in multiple energy phases with a specific set of parameters representing the estimated “energy  
6155 footprint” of an appliance. The data carried in the Power Profile can be updated during the different states of a Power  
6156 Profile; since it represents a forecast of energy, duration and peak power of energy phases, it SHALL be considered  
6157 as an estimation and not derived by measurements.

6158 The Power Profile MAY also be used by an energy management system, together with other specific interfaces  
6159 supported by the device, in order to schedule and control the device operation and to perform energy management  
6160 within a home network. For more informative examples on how the Power Profile cluster might be used, see Chapter  
6161 15 Appliance Management, section 3.

6162

Figure 3-67. Typical Usage of the Power Profile Cluster



6163

Note: Device names are examples for illustration purposes only

6164 

### 3.17.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

6165 

### 3.17.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	PWR

6166 

### 3.17.1.3 Cluster Identifiers

Identifier	Name
0x001a	Power Profile

6167 

## 3.17.2 References

6168 The following standards and specifications contain provisions, which through reference in this document constitute  
 6169 provisions of this specification. All the standards and specifications listed are normative references. At the time of  
 6170 publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to  
 6171 agreements based on this specification are encouraged to investigate the possibility of applying the most recent  
 6172 editions of the standards and specifications indicated below.

6173 

## 3.17.3 General Description

6174 

### 3.17.3.1 Dependencies

6175 The Power Profile Cluster is dependent upon the Appliance Control Cluster for the parts regarding the status  
 6176 notification and power management commands. Other specific clusters for actuation for devices different than Smart  
 6177 Appliances. Due to the possible length of the Power Profile commands, the devices supporting the Power Profile  
 6178 cluster MAY leverage on Partitioning if required by the application.



6179 **3.17.4 Server Attributes**

6180 . The following attributes represent the parameters for each Power Profile’s phases.

6181 **Table 3-135. Attributes of the Power Profile Cluster**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Default</b>	<b>M/O</b>
0x0000	<i>TotalProfileNum</i>	uint8	0x01 to 0xfe	R	1	M
0x0001	<i>MultipleScheduling</i>	bool	0x00 to 0x01	R	FALSE	M
0x0002	<i>EnergyFormatting</i>	map8	0x00 to 0xff	R	0x01*	M
0x0003	<i>EnergyRemote</i>	bool	TRUE or FALSE	R	FALSE	M
0x0004	<i>ScheduleMode</i>	map8	0x00 to 0xff	RWP	0x00	M

6182

6183 \* 1/10 of Watt Hours represented

6184 **3.17.4.1 TotalProfileNum Attribute**

6185 The *TotalProfileNum* attribute represents the total number of profiles supported by the device. The minimum value  
 6186 for this attribute SHALL be 1.

6187 **3.17.4.2 MultipleScheduling Attribute**

6188 The *MultipleScheduling* attribute specifies if the server side of the Power Profile cluster supports the scheduling of  
 6189 multiple Energy Phases or it does support the scheduling of a single energy phase of the Power Profile at a time. If  
 6190 more than a single energy phases MAY be scheduled simultaneously the *MultipleScheduling* attribute SHALL be set  
 6191 to TRUE. In this case the device supporting the Power Profile server SHALL be able to process and manage scheduling  
 6192 commands carrying the schedule of more than one energy phase.

6193 If the *MultipleScheduling* attribute is FALSE the device supporting the Power Profile client (e.g., EMS) SHALL be  
 6194 allowed to schedule just a single energy phase.

6195 **3.17.4.3 EnergyFormatting Attribute**

6196 The *EnergyFormatting* attribute provides a method to properly decipher the number of digits and the decimal location  
 6197 of the values found in the Energy Fields carried by the Power Profile Notification and Power Profile Response  
 6198 commands. This attribute is to be decoded as follows:

- 6199 • Bits 0 to 2: Number of Digits to the right of the Decimal Point.
- 6200 • Bits 3 to 6: Number of Digits to the left of the Decimal Point.
- 6201 • Bit 7: If set, suppress leading zeros.

6202 This attribute SHALL be used against the Energy fields.

6203 **3.17.4.4 EnergyRemote Attribute**

6204 The *EnergyRemote* attribute indicates whether the power profile server (e.g., appliance) is configured for remote  
 6205 control (e.g., by an energy management system). This refers to the selection chosen by the user on the remote control  
 6206 feature of the device. If the value is FALSE, the remote energy management is disabled, otherwise it is enabled. If the  
 6207 EnergyRemote is equal to FALSE all the supported PowerProfile SHALL set the Power Profile Remote Control field  
 6208 in the PowerProfile record equal to FALSE.

6209 If the *EnergyRemote* attribute value is equal to TRUE, at least one PowerProfile SHALL be remotely controllable  
6210 setting the Power Profile Remote Control field in the PowerProfile record to TRUE.

6211 **Table 3-136. *EnergyRemote* Attribute**

Energy Remote Value	Description
0x00	FALSE = Remote Energy Management disabled
0x01	TRUE = Remote Energy Management enabled

### 6212 3.17.4.5 ScheduleMode Attribute

6213 The *ScheduleMode* attribute describes the criteria that SHOULD be used by the Power Profile cluster client side (e.g.,  
6214 energy management system) to schedule the power profiles.

#### 6215 3.17.4.5.1 Schedule Mode Field BitMap

6216 **Table 3-137. *ScheduleMode* Attribute**

Bit	Description
bit0	bit0=1 : Schedule Mode Cheapest
bit1	bit1 =1: Schedule Mode Greenest
bit2 to bit7	Reserved

6217 If the *ScheduleMode* attribute is set to the value 0x00, the scheduling criteria is demanded to the Power Profile cluster  
6218 client side, which means that no specific preferences on the schedule mode are requested by the device supporting the  
6219 server side of the power Profile cluster.

6220 If “Schedule Mode Cheapest” is selected then the energy management system SHALL try to schedule the Power  
6221 Profile to minimize the user’s energy bill.

6222 If “Schedule Mode Greenest” is selected then the energy management system SHALL try to schedule the Power  
6223 Profile to provide the highest availability of renewable energy sources.

6224 Please note that how the energy management system MAY obtain “cheapest” or “greenest” information and estimate  
6225 scheduling times is out of scope of this specification.

6226 If more than a single bit is selected in the *ScheduleMode* bitmask, the Power Profile client SHALL try to calculate the  
6227 schedule following all the selected criteria.

6228 If all the bits are set to zero not specific optimization metrics preferences are requested by the device supporting the  
6229 Power Profile server.

## 6230 3.17.5 Server Commands Received

6231 The command IDs for the commands received by the server side of the Power Profile Cluster are listed in Table 3-138.

6232 **Table 3-138. Cluster-Specific Commands Received by the Server**

Command Identifier Field Value	Description	M/O
0x00	<i>PowerProfileRequest</i>	M

Command Identifier Field Value	Description	M/O
0x01	<i>PowerProfileStateRequest</i>	M
0x02	<i>GetPowerProfilePriceResponse</i>	M
0x03	<i>GetOverallSchedulePriceResponse</i>	M
0x04	<i>EnergyPhasesScheduleNotification</i>	M
0x05	<i>EnergyPhasesScheduleResponse</i>	M
0x06	<i>PowerProfileScheduleConstraintsRequest</i>	M
0x07	<i>EnergyPhasesScheduleStateRequest</i>	M
0x08	<i>GetPowerProfilePriceExtendedResponse</i>	M

6233 **3.17.5.1 PowerProfileRequest Command**

6234 The *PowerProfileRequest* Command is generated by a device supporting the client side of the Power Profile cluster  
 6235 in order to request the Power Profile of a server device. It is possible to request all profiles (without knowing how  
 6236 many Power Profiles the server has) or to request a specific *PowerProfileID*.

6237 In the case of multiple profiles the server SHOULD send multiple messages, one for each Power Profile.

6238 Although the profile is in a Power Profile running state (see *PowerProfileState*), the Power Profile Response  
 6239 transmitted as a reply to a *PowerProfileRequest* command SHALL carry all the energy phases of the estimated Power  
 6240 Profile, including the previous energy phases and the current energy phase which is running. The parameters of the  
 6241 Power Profile (e.g., the *ExpectedDuration* or the *Energy* fields of all the energy phases) MAY be updated for the same  
 6242 Power Profile due to a tuning in the forecast.

6243 **3.17.5.1.1 Payload Format**

6244 The *PowerProfileRequest* Command payload SHALL be formatted as illustrated in Figure 3-68.

6245 **Figure 3-68. Format of the *PowerProfileRequest* Command Payload**

Octets	1
Data Type	uint8
Field Name	PowerProfileID

6246 **3.17.5.1.1.1 Payload Details**

6247 The payload of the *PowerProfileRequest* command carries the fields defined in Figure 3-68.

6248 The *PowerProfileID* field specifies which profile (in the range 1 to *TotalProfileNum*) is requested. The special value  
 6249 0x00 of this field does not refer to a particular profile; if 0x00 value is received the device SHOULD send details  
 6250 related to all the available profiles.

6251 The *PowerProfileID* field SHALL NOT be greater than *TotalProfileNum*.

6252 **3.17.5.1.2 When Generated**

6253 This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to  
6254 request the power profile to a device supporting the Power Profile cluster server side (e.g., White Good).

6255 **3.17.5.1.3 Effect on Receipt**

6256 The device that receives the Power Profile Request command SHALL reply with a *PowerProfileResponse* if  
6257 supported. If the command is not supported the device SHALL reply with a standard ZCL Default response with status  
6258 UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification).

6259 If the requested profile data are not available, the device SHALL reply with a standard ZCL response NOT\_FOUND  
6260 0x8b (according to ZCL specification).

6261 **3.17.5.2 PowerProfileStateRequest Command**

6262 The *PowerProfileStateRequest* command is generated in order to retrieve the identifiers of current Power Profiles.  
6263 This command does not have a payload.

6264 **3.17.5.2.1 Effect on Receipt**

6265 On receipt of this command, the device SHALL generate a *PowerProfileStateResponse* command.

6266 **3.17.5.3 GetPowerProfilePriceResponse Command**

6267 The *GetPowerProfilePriceResponse* command allows a device (client) to communicate the cost associated with a  
6268 defined Power Profile to another device (server) requesting it. If the Price information requested related to the Power  
6269 Profile is not available yet the response SHALL be a ZCL default response with "NOT FOUND" Status.

6270 **3.17.5.3.1 Payload Format**

6271 The *GetPowerProfilePriceResponse* command payload SHALL be formatted as illustrated in Figure 3-69.

6272 **Figure 3-69. Format of the *GetPowerProfilePriceResponse* Command**

<b>Octets</b>	1	2	4	1
<b>Data Type</b>	uint8	uint16	uint32	uint8
<b>Field Name</b>	Power Profile ID	Currency	Price	Price Trailing Digit

6273 **3.17.5.3.1.1 Payload Details**6274 **PowerProfileID**

6275 The PowerProfileID field represents the identifier of the specific profile described by the Power Profile.

6276 This is typically a sequential and contiguous number ranging from 1 to *TotalProfileNum*.

6277 **Currency**

6278 The Currency field identifies the local unit of currency used in the price field. This field is thought to be useful for  
6279 displaying the appropriate symbol for a currency (i.e., \$, €). The value of the currency field SHOULD match the values  
6280 defined by ISO 4217.

6281 **Price**

6282 The Price field contains the price of the energy of a specific Power Profile measured in base unit of Currency per Unit  
 6283 of Measure (as described in the Metering Cluster, see SE specification) with the decimal point located as indicated by  
 6284 the PriceTrailingDigit field when the energy is delivered to the premise.

6285 **Price Trailing Digit**

6286 The PriceTrailingDigit field determines where the decimal point is located in the price field. The PriceTrailingDigit  
 6287 indicates the number of digits to the right of the decimal point.

6288 **3.17.5.3.2 When Generated**

6289 This command is generated when the command Get Power Profile Price is received. Please refer to Get Power Profile  
 6290 Price command description.

6291 **3.17.5.3.3 Effect on Receipt**

6292 On receipt of this command, the originator (server) is notified of the associated cost of the requested Power Profile,  
 6293 calculated by the client side of the Power Profile (see 9.7.10.1 for sequence diagrams and examples).

6294 **3.17.5.4 GetOverallSchedulePriceResponse Command**

6295 The *GetOverallSchedulePriceResponse* command allows a client to communicate the overall cost associated to all  
 6296 Power Profiles scheduled to a server requesting it. If the Price information requested is not available the response  
 6297 SHALL be a ZCL default response with “NOT FOUND” Status. The overall cost provided by the Power Profile Client  
 6298 side (e.g., energy management system) is intended as the cost of all the scheduled power profiles. This information  
 6299 MAY be helpful to assess the overall benefit provided by the scheduler, since a change in the scheduling of a specific  
 6300 device might -in some cases-increase its associated Power Profile cost. In fact in that case the schedule SHALL provide  
 6301 a global optimization by reducing the overall cost of all the scheduled power profiles, then reducing the energy bill  
 6302 for the user.

6303 **3.17.5.4.1 Payload Format**

6304 The Get Overall Schedule Price Response command payload SHALL be formatted as illustrated in Figure 3-70.

6305 **Figure 3-70. Format of the *GetOverallSchedulePriceResponse* Command**

<b>Octets</b>	2	4	1
<b>Data Type</b>	uint16	uint32	uint8
<b>Field Name</b>	Currency	Price	Price Trailing Digit

6306 **3.17.5.4.2 Payload Details**

6307 See *GetPowerProfilePriceResponse* command payload details.

6308 **3.17.5.4.3 When Generated**

6309 This command is generated when the command *GetOverallSchedulePriceRequest* is received.

6310 **3.17.5.4.4 Effect on Receipt**

6311 On receipt of this command, the originator is notified of the overall cost of the scheduled Power Profiles, calculated  
6312 by the Power Profile cluster client side. This information MAY be used to assess the overall benefit provided by the  
6313 scheduler, which might be dependent on the schedule constraints. For more information, see Chapter 15 Appliance  
6314 Management, section 3.

6315 **3.17.5.5 Energy Phases Schedule Notification Command**

6316 The Energy Phases Schedule Notification command is generated by a device supporting the client side of the Power  
6317 Profile cluster in order to schedule the start of a Power Profile and its energy phases (they MAY be more than one in  
6318 case of *MultipleScheduling* attribute equal to TRUE) on a the device supporting the server side of the Power Profile  
6319 cluster, which did not solicit the schedule (“un-solicited” schedule). That happens when the Power Profile State carries  
6320 a *PowerProfileRemoteControl* field equal to TRUE and the Energy Phase has a *MaxActivationDelay* different than  
6321 0x0000 (please note that changes on an already scheduled energy phase or power profile are possible but SHOULD  
6322 be applied just in case of sensible advantages). The mechanisms designed to find the proper schedule are not part of  
6323 the description of this command.

6324 Please consider that, in case the *MultipleScheduling* attribute is FALSE (which means that the server side of the Power  
6325 Profile cluster SHALL support the schedule of only a single energy phase at once), the Energy Phases Schedule  
6326 Notification command SHOULD also be used to set a pause between two energy phases (energy pause behavior). In  
6327 this case the Power Profile State MAY have any values but the command SHALL be issued only if the  
6328 *PowerProfileRemoteControl* is set to TRUE and the Energy Phase has a *MaxActivationDelay* different than 0x0000.

6329 **3.17.5.5.1 Payload Format**

6330 The Energy Phases Schedule Notification command payload SHALL be formatted as illustrated in Figure 3-71.

6331 **Figure 3-71. Format of the *EnergyPhasesScheduleNotification* Command Payload**

<b>Octets</b>	1	1	1	2	...	1	2
<b>Data Type</b>	uint8	uint8	uint8	uint16	...	uint8	uint16
<b>Field Name</b>	Power ProfileID	Num of Scheduled Phases	Energy PhaseID <sub>n</sub>	Scheduled Time <sub>n</sub>	...	Energy PhaseID <sub>n</sub>	Scheduled Time <sub>n</sub>

6332 **3.17.5.5.1.1 Payload Details**

6333 The payload of the *EnergyPhasesScheduleNotification* command carries the fields defined in Figure 3-71. Each  
6334 *EnergyPhasesScheduleNotification* message SHALL include only one Power Profile and the energy phases of that  
6335 Power Profile that needs to be scheduled. In case this command needs to be sent to a device supporting the server side  
6336 of the power Profile Cluster with the *MultipleScheduling* attribute set to false, the payload of  
6337 *EnergyPhasesScheduleNotification* command SHALL carry just one phase and the Scheduled Time field SHALL  
6338 indicate the time scheduled for the whole Power Profile to start (in case the Power Profile is not started yet). If the  
6339 Power Profile is in ENERGY\_PHASE\_RUNNING state and the server side of the cluster has the *MultipleScheduling*  
6340 attribute set to false, the *EnergyPhasesScheduleNotification* command SHALL carry the scheduled time of the next  
6341 energy phase.

6342 **PowerProfileID**

6343 See definition in *PowerProfileNotification* command.

6344 **Num of Scheduled Phases**

6345 The Num of Scheduled Phases field represents the total number of the energy phases of the Power Profile that need to  
6346 be scheduled by this command.

6347 The Energy phases that are not required to be scheduled SHALL NOT be counted in Num of Scheduled Phases field.  
6348 The Num of Scheduled Phases SHALL be equal to 1 in case the *MultipleScheduling* attribute set to FALSE (only one  
6349 energy phase SHALL be scheduled at a time). The *Num of Scheduled Phases* MAY be greater than 1 in case the  
6350 *MultipleScheduling* attribute set to TRUE (scheduling of multiple energy phases at the same time).

### 6351 **EnergyPhaseID**

6352 See definition in *PowerProfileNotification* command.

### 6353 **Scheduled Time**

6354 The Scheduled Time field represents the relative time scheduled in respect to the end of the previous energy phase.  
6355 The unit is the minute. The Scheduled Time for the first Energy phase represents the scheduled time (expressed in  
6356 relative encoding in respect to the current time) for the start of the Power Profile. The Scheduled Time fields for the  
6357 subsequent Energy phases represent the relative time in minutes in respect to the previous scheduled Energy phase.  
6358 The Energy phases that are not required to be scheduled will not be included in the commands and not be counted in  
6359 Num of Scheduled Phases field. Only the Power Profile carrying a Power Profile Remote Control field equal to TRUE  
6360 (as indicated in Power Profile State Notification command) and the Energy Phases supporting *MaxActivationDelay*  
6361 different than 0x0000 SHALL be schedulable (as indicated in Power Profile Notification command).

### 6362 **3.17.5.5.2 When Generated**

6363 This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), has  
6364 calculated a specific schedule for a Power Profile and needs to send the schedule (i.e., “unsolicited” schedule) to a  
6365 device supporting the Power Profile cluster server side (e.g., White Goods). This command SHALL be generated only  
6366 if the recipient devices support schedulable Power Profiles (i.e., only if the Power Profile carries the first Energy Phase  
6367 with a *MaxActivationDelay* different than 0x0000).

### 6368 **3.17.5.5.3 Effect on Receipt**

6369 The device that receives the *EnergyPhasesScheduleNotification* command SHALL reply with a standard Default  
6370 response only if requested in the ZCL header of the *EnergyPhasesScheduleNotification* command or there is an error  
6371 (as from ZCL specification).

6372 If the device that receives the *EnergyPhasesScheduleNotification* command cannot schedule the energy phases  
6373 because the activation delay of any of carried phases is equal to zero, it SHALL reply with a standard Default response  
6374 with the error code NOT\_AUTHORIZED (0x7e).

6375 In case the scheduling state of the recipient entity changes after the reception of this command, the recipient will issue  
6376 an Energy Phases Schedule State Notification.

### 6377 **3.17.5.6 EnergyPhasesScheduleResponse Command**

6378 This command is generated by the client side of Power Profile cluster as a reply to the *EnergyPhasesScheduleRequest*  
6379 command.

### 6380 **3.17.5.6.1 Payload Format**

6381 The *EnergyPhasesScheduleResponse* command payload SHALL have the same payload as  
6382 *EnergyPhasesScheduleNotification* command (*EnergyPhasesScheduleNotification* command, but “solicited”  
6383 schedule because it is triggered by the *EnergyPhasesScheduleRequest* command). For more information, see Chapter  
6384 15, Appliance Management, section 3.

### 6385 **3.17.5.6.1.1 Payload Details**

6386 The payload of the *EnergyPhasesScheduleResponse* command carries the same fields as the  
6387 *EnergyPhasesScheduleNotification* command. (*EnergyPhasesScheduleNotification* command, but “solicited”  
6388 schedule because it is triggered by the *EnergyPhasesScheduleRequest* command)

**6389 3.17.5.6.2 When Generated**

6390 This command is generated when the server side of the Power Profile cluster (e.g., a White Goods device), has  
6391 requested, using the *EnergyPhasesScheduleRequest*, the schedule of a specific power profile to a device supporting  
6392 the Power Profile cluster client side (e.g., Home gateway) which SHALL calculate the schedules (“solicited” schedule)  
6393 and reply with the *EnergyPhasesScheduleResponse*.

**6394 3.17.5.6.3 Effect on Receipt**

6395 The device that receives the *EnergyPhasesScheduleResponse* command SHALL reply with a standard Default  
6396 response only if requested in the ZCL header of the *EnergyPhasesScheduleResponse* command. If the reception of  
6397 *EnergyPhasesScheduleResponse* command is not supported the device SHALL reply with a standard ZCL Default  
6398 response with status UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification).

6399 In case the scheduling state of the recipient entity changes after the reception of this command, the recipient will issue  
6400 an *EnergyPhasesScheduleStateNotification*.

**6401 3.17.5.7 PowerProfileScheduleConstraintsRequest Command**

6402 The *PowerProfileScheduleConstraintsRequest* command is generated by client side of the Power Profile cluster in  
6403 order to request the constraints of the Power Profile of a server, in order to set the proper boundaries for the scheduling  
6404 when calculating the schedules.

**6405 3.17.5.7.1 Payload Format**

6406 The *PowerProfileScheduleConstraintsRequest* command payload is the same as the one used for *PowerProfileRequest*  
6407 command. For more information, see Chapter 15, Appliance Management, section 3.

**6408 3.17.5.7.1.1 Payload Details**

6409 The payload of the *PowerProfileScheduleConstraintsRequest* command carries the fields defined in  
6410 *PowerProfileRequest* command.

6411 The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the constraints are  
6412 referring to.

**6413 3.17.5.7.2 When Generated**

6414 This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to  
6415 request the constraints of the power profile to a device supporting the Power Profile cluster server side (e.g.,  
6416 Whitegood).

**6417 3.17.5.7.3 Effect on Receipt**

6418 The device that receives the Power Profile Schedule Constraints Request command SHALL reply with a Power Profile  
6419 Schedule Constraints Response if supported. If the command is not supported, the device SHALL reply with a standard  
6420 ZCL Default response with status UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification).

6421 If the requested profile data are not available, the device SHALL reply with a standard ZCL response NOT\_FOUND  
6422 0x8b (according to ZCL specification).



### 6423 **3.17.5.8 EnergyPhasesScheduleStateRequest Command**

6424 The *EnergyPhasesScheduleStateRequest* command is generated by a device supporting the client side of the Power  
6425 Profile cluster to check the states of the scheduling of a power profile, which is supported in the device implementing  
6426 the server side of Power Profile cluster. This command can be used to re-align the schedules between server and client  
6427 (e.g., after a client reset).

#### 6428 **3.17.5.8.1 Payload Format**

6429 The *EnergyPhasesScheduleStateRequest* command payload is the same as the one used for *PowerProfileRequest*  
6430 command. For more information, see Chapter 15, Appliance Management, section 3.

##### 6431 **3.17.5.8.1.1 Payload Details**

6432 The payload of the *EnergyPhasesScheduleStateRequest* command carries the same fields defined in the  
6433 *PowerProfileRequest* command. For more information, see Chapter 15, Appliance Management, section 3.

6434 The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the constraints are  
6435 referring to.

#### 6436 **3.17.5.8.2 When Generated**

6437 This command is generated when the client side of the Power Profile cluster (e.g., a Home gateway device), needs to  
6438 check the schedules of the Power Profile to a device supporting the Power Profile cluster server side (e.g., White  
6439 Good).

#### 6440 **3.17.5.8.3 Effect on Receipt**

6441 The server that receives the *EnergyPhasesScheduleStateRequest* command SHALL reply to the client with an  
6442 *EnergyPhasesScheduleStateResponse*, if supported. If the command is not supported, the servers SHALL reply with  
6443 a standard ZCL Default response with status UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification).

6444 If the requested profile data are not available (e.g., invalid Power Profile ID), the server SHALL reply with a standard  
6445 ZCL response NOT\_FOUND 0x8b (according to ZCL specification).

6446 If the server does not have any schedules set, it SHALL reply with a *EnergyPhasesScheduleStateResponse* carrying  
6447 *NumofScheduledPhases* equal to zero (see Format of the *EnergyPhasesScheduleStateResponse* in case of no scheduled  
6448 phases).

### 6449 **3.17.5.9 GetPowerProfilePriceExtendedResponse Command**

6450 The *GetPowerProfilePriceExtendedResponse* command allows a device (client) to communicate the cost associated  
6451 to all Power Profiles scheduled to another device (server) requesting it according to the specific options contained in  
6452 the *EnergyPhasesScheduleStateResponse*. If the Price information requested is not available, the response SHALL be  
6453 a ZCL default response with "NOT FOUND" Status.

#### 6454 **3.17.5.9.1 Payload Format**

6455 The *EnergyPhasesScheduleStateResponse* command payload SHALL be formatted as the  
6456 *GetPowerProfilePriceResponse* command.

#### 6457 **3.17.5.9.2 Payload Details**

6458 See *GetPowerProfilePriceResponse* command payload details.

6459 **3.17.5.9.3 When Generated**6460 This command is generated when the command *GetPowerProfilePriceExtendedResponse* is received.6461 **3.17.5.9.4 Effect on Receipt**

6462 On receipt of this command, the originator is notified of cost of the scheduled Power Profiles, calculated by the Power  
 6463 Profile cluster server side according to the specific option transmitted in the *EnergyPhasesScheduleStateResponse*  
 6464 command (e.g., cost at specific *PowerProfileStartTime*). For more information, see Chapter 15, Appliance  
 6465 Management, section 3.

6466 **3.17.6 Server Commands Generated**

6467 Cluster-specific commands are generated by the server, as shown in Table 3-139.

6468 **Table 3-139. Cluster-Specific Commands Sent by the Server**

Command Identifier Field Value	Description	M/O
0x00	<i>PowerProfileNotification</i>	M
0x01	<i>PowerProfileResponse</i>	M
0x02	<i>PowerProfileStateResponse</i>	M
0x03	<i>GetPowerProfilePrice</i>	O
0x04	<i>PowerProfilesStateNotification</i>	M
0x05	<i>GetOverallSchedulePrice</i>	O
0x06	<i>EnergyPhasesScheduleRequest</i>	M
0x07	<i>EnergyPhasesScheduleStateResponse</i>	M
0x08	<i>EnergyPhasesScheduleStateNotification</i>	M
0x09	<i>PowerProfileScheduleConstraintsNotification</i>	M
0x0A	<i>PowerProfileScheduleConstraintsResponse</i>	M
0x0B	<i>GetPowerProfilePriceExtended</i>	O

6469

6470 **3.17.6.1 PowerProfileNotification Command**

6471 The *PowerProfileNotification* command is generated by a device supporting the server side of the Power Profile  
 6472 cluster in order to send the information of the specific parameters (such as Peak power and others) belonging to each  
 6473 phase.

6474 **3.17.6.1.1 Payload Format**6475 The *PowerProfileNotification* command payload SHALL be formatted as illustrated in Figure 3-72.

6476 **Figure 3-72. Format of the *PowerProfileNotification* Command Payload (1 of 2)**

<b>Octets</b>	1	1	1	1	1	2	2	2
<b>Data Type</b>	uint8	uint8	uint8	uint8	uint8	uint16	uint16	uint16
<b>Field Name</b>	<i>Total Profile Num</i>	<i>Power ProfileID</i>	<i>Num of Transferred Phases</i>	<i>Energy PhaseID<sub>1</sub></i>	<i>Macro PhaseID<sub>1</sub></i>	<i>ExpectedDuration<sub>1</sub></i>	<i>Peak Power<sub>1</sub></i>	<i>Energy<sub>1</sub></i>

6477

<b>Octets</b>	2	...	1	1	2	2	2	2
<b>Data Type</b>	uint16	...	uint8	uint8	uint16	uint16	uint16	uint16
<b>Field Name</b>	<i>MaxActivationDelay<sub>1</sub></i>	...	<i>Energy PhaseID<sub>n</sub></i>	<i>Macro PhaseID<sub>n</sub></i>	<i>Expected Duration<sub>n</sub></i>	<i>Peak Power<sub>n</sub></i>	<i>Energy<sub>n</sub></i>	<i>MaxActivationDelay<sub>n</sub></i>

6478 **3.17.6.1.1.1 Payload Details**

6479 The payload of the *PowerProfileNotification* command carries the fields defined in Figure 3-72. Each  
 6480 *PowerProfileNotification* message SHALL include only one Power Profile.

6481 If multiple phases are transferred within a single *PowerProfileNotification* command (i.e., *Number of Transferred*  
 6482 *Phases* greater than 1), the parameters of the other phases (*PhaseID*, *ExpectedDuration*, etc.) SHOULD be carried in  
 6483 the payload. Each phase has a fixed number of parameters and the total length is 10 octets, so that the total length of  
 6484 the payload could be calculated with the following formula:

6485 
$$\text{Total Payload Length} = 1 + 1 + 1 + (\text{Num of Transferred Phases} * 10)$$

6486 **TotalProfileNum**

6487 For more information, see Chapter 15, Appliance Management, section 3.

6488 **PowerProfileID**

6489 The PowerProfileID field represents the identifier of the specific profile described by the Power Profile.

6490 This field contains a sequential and contiguous number ranging from 1 to *TotalProfileNum*.

6491 **Num of Transferred Phases**

6492 This field represents the number of the energy phases of the Power Profile.

6493 **MacroPhaseID**

6494 The MacroPhaseID field represents the identifier of the specific phase (operational-displayed) described by the Power  
 6495 Profile.

6496 This reference could be used in conjunction with a table of ASCII strings, describing the label of the functional phase.  
 6497 This table is not described in the context of the Power Profile because it MAY be not functionally linked with energy  
 6498 management.

6499 **EnergyPhaseID**

6500 The EnergyPhaseID field indicates the identifier of the specific energy phase described by the Power Profile.

6501 This is a sequential and contiguous number ranging from 1 to the maximum number of phases belonging to the Power  
6502 Profile.

6503 The value 0xFF SHALL be used to specify invalid energy phase (e.g., for a Power Profile in IDLE state).

#### 6504 **ExpectedDuration**

6505 The ExpectedDuration field represents the estimated duration of the specific phase. Each unit is a minute.

#### 6506 **PeakPower**

6507 The PeakPower field represents the estimated power for the specific phase. Each unit is a Watt.

#### 6508 **Energy**

6509 The Energy field represents the estimated energy consumption for the accounted phase. Each unit is Watt per hours,  
6510 according to the formatting specified in the *EnergyFormatting* attribute. For more information, see Chapter 15,  
6511 Appliance Management, section 3. The Energy value fulfills the following equation:

6512  $Energy \leq PeakPower(Watt) * ExpectedDuration(sec)$

#### 6513 **MaxActivationDelay**

6514 The MaxActivationDelay field indicates the maximum interruption time between the end of the previous phase and  
6515 the beginning of the specific phase. Each unit is a minute.

6516 The special value 0x0000 means that it is not possible to insert a pause between the two consecutive phases.

6517 The MaxActivationDelay field of the first energy phase of a Power Profile SHALL be set to the value 0xFFFF.

### 6518 **3.17.6.1.2 When Generated**

6519 This command is generated when the server side of the Power Profile cluster (e.g., a White Good device), need to  
6520 send the representation of its power profile to a controller device supporting the Power Profile cluster client side (e.g.,  
6521 Home Gateway).

### 6522 **3.17.6.1.3 Effect on Receipt**

6523 The device that receives the *PowerProfileNotification* command SHALL reply with a standard Default response if  
6524 requested in the ZCL header of the *PowerProfileNotification* command.

## 6525 **3.17.6.2 PowerProfileResponse Command**

6526 This command is generated by the server side of Power Profile cluster as a reply to the *PowerProfileRequest* command.  
6527 If the reception of *PowerProfileRequest* command is not supported the device SHALL reply with a standard ZCL  
6528 Default response with status UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification).

6529 If the profile data requested are not available, the device SHALL reply with a standard ZCL response  
6530 INVALID\_VALUE 0x87 (as ZCL specification).

### 6531 **3.17.6.2.1 Payload Format**

6532 The *PowerProfileResponse* Command payload SHALL be formatted as illustrated in Figure 3-72 (same as  
6533 *PowerProfileNotification* command).

#### 6534 **3.17.6.2.1.1 Payload Details**

6535 The payload of the *PowerProfileResponse* command carries the fields defined in Figure 3-72 (the same as  
6536 *PowerProfileNotification* command).

6537 **3.17.6.2.2 When Generated**

6538 This command is generated by the server side of Power Profile cluster (e.g., White Good) as a reply to the  
 6539 *PowerProfileRequest* command sent by the client side (e.g., a Home gateway device).

6540 **3.17.6.2.3 Effect on Receipt**

6541 The device that receives the *PowerProfileResponse* command SHALL reply with a standard Default response if  
 6542 requested in the ZCL header of the *PowerProfileResponse* command.

6543 The device that receives the *PowerProfileResponse* command SHALL reply with a standard ZCL Default response  
 6544 with status UNSUP\_CLUSTER\_COMMAND 0x81 (as from ZCL specification) if the reception of this command is  
 6545 not supported.

6546 If the profile data requested are not available, the device SHALL reply with a standard ZCL response  
 6547 INVALID\_VALUE 0x87 (as ZCL specification).

6548 **3.17.6.3 PowerProfileStateResponse Command**

6549 The *PowerProfileStateResponse* command allows a device (server) to communicate its current Power Profile(s) to  
 6550 another device (client) that previously requested them.

6551 **3.17.6.3.1 Payload Format**

6552 The *PowerProfileStateResponse* command payload SHALL be formatted as illustrated in Figure 3-73.

6553 **Figure 3-73. Format of the *PowerProfileStateResponse* Command Frame**

<b>Octets</b>	1	4	4	...	4
<b>Field Name</b>	Power Profile Count	Power Profile Record 1	Power Profile Record 2	...	Power Profile Record <i>n</i>

6554  
 6555 Each Power Profile record SHALL be formatted as illustrated in Figure 3-74.

6556 **Figure 3-74. Format of the Power Profile Record Field**

<b>Octets</b>	1	1	1	1
<b>Data Type</b>	uint8	uint8	bool	enum8
<b>Field Name</b>	Power Profile ID	Energy Phase ID	PowerProfile RemoteControl	PowerProfile State

6557 **3.17.6.3.1.1 Payload Details**

6558 **Power Profile Count**

6559 The Power Profile Count is the number of Power Profile Records that follow in the message.

6560 **Power Profile Record**

6561 The Power Profile record supports the following fields:

- 6562 • **Power Profile ID:** The identifier of the Power Profile as requested.

- 6563 • **Energy Phase ID:** The current Energy Phase ID of the specific Profile ID; this value SHALL be set to invalid  
6564 0xFF when PowerProfileState indicates a Power Profile in POWER\_PROFILE\_IDLE state.
- 6565 • **PowerProfileRemoteControl:** It indicates if the PowerProfile is currently remotely controllable or not; if the  
6566 Power Profile is not remotely controllable it cannot be scheduled by a Power Profile client.
- 6567 • **PowerProfileState:** An enumeration field representing the current state of the Power Profile (see Table 3-140).

6568

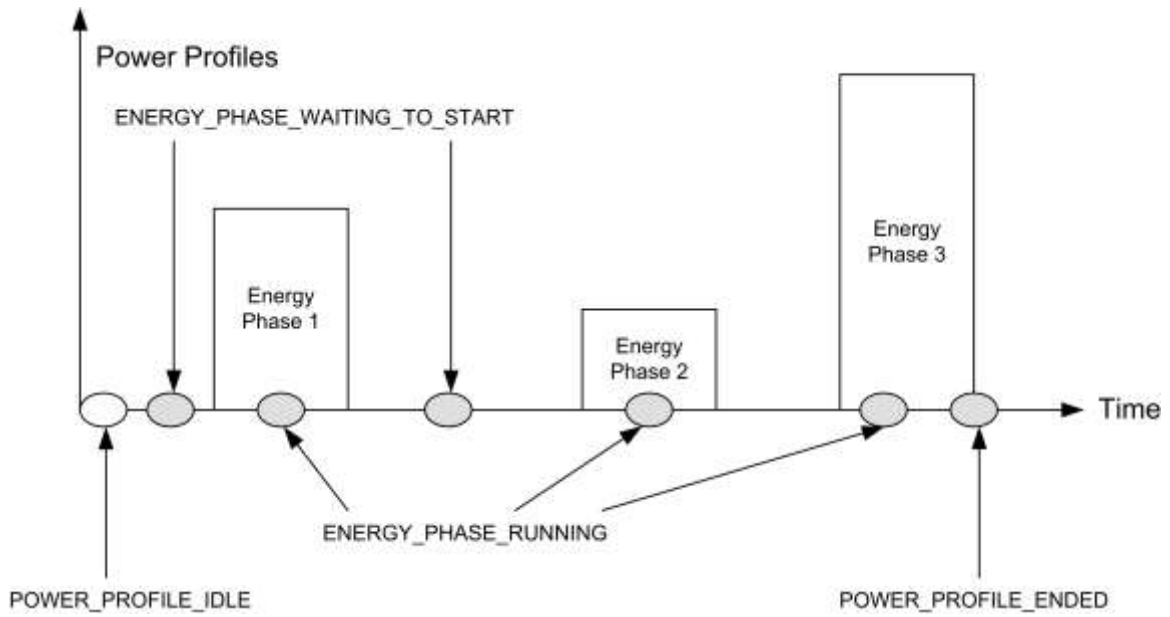
**Table 3-140. PowerProfileState Enumeration Field**

Enumeration	Value	Description
POWER_PROFILE_IDLE	0x00	The PP is not defined in its parameters.
POWER_PROFILE_PROGRAMMED	0x01	The PP is defined in its parameters but without a scheduled time reference
ENERGY_PHASE_RUNNING	0x03	An energy phase is running
ENERGY_PHASE_PAUSED	0x04	The current energy phase is paused
ENERGY_PHASE_WAITING_TO_START	0x05	The Power Profile is in between two energy phases (one ended, the other not yet started). If the first Energy Phase is considered, this state indicates that the whole power profile is not yet started, but it has been already programmed to start
ENERGY_PHASE_WAITING_PAUSED	0x06	The Power Profile is set to Pause when being in the ENERGY_PHASE_WAITING_TO_START state.
POWER_PROFILE_ENDED	0x07	The whole Power Profile is terminated

6569

6570

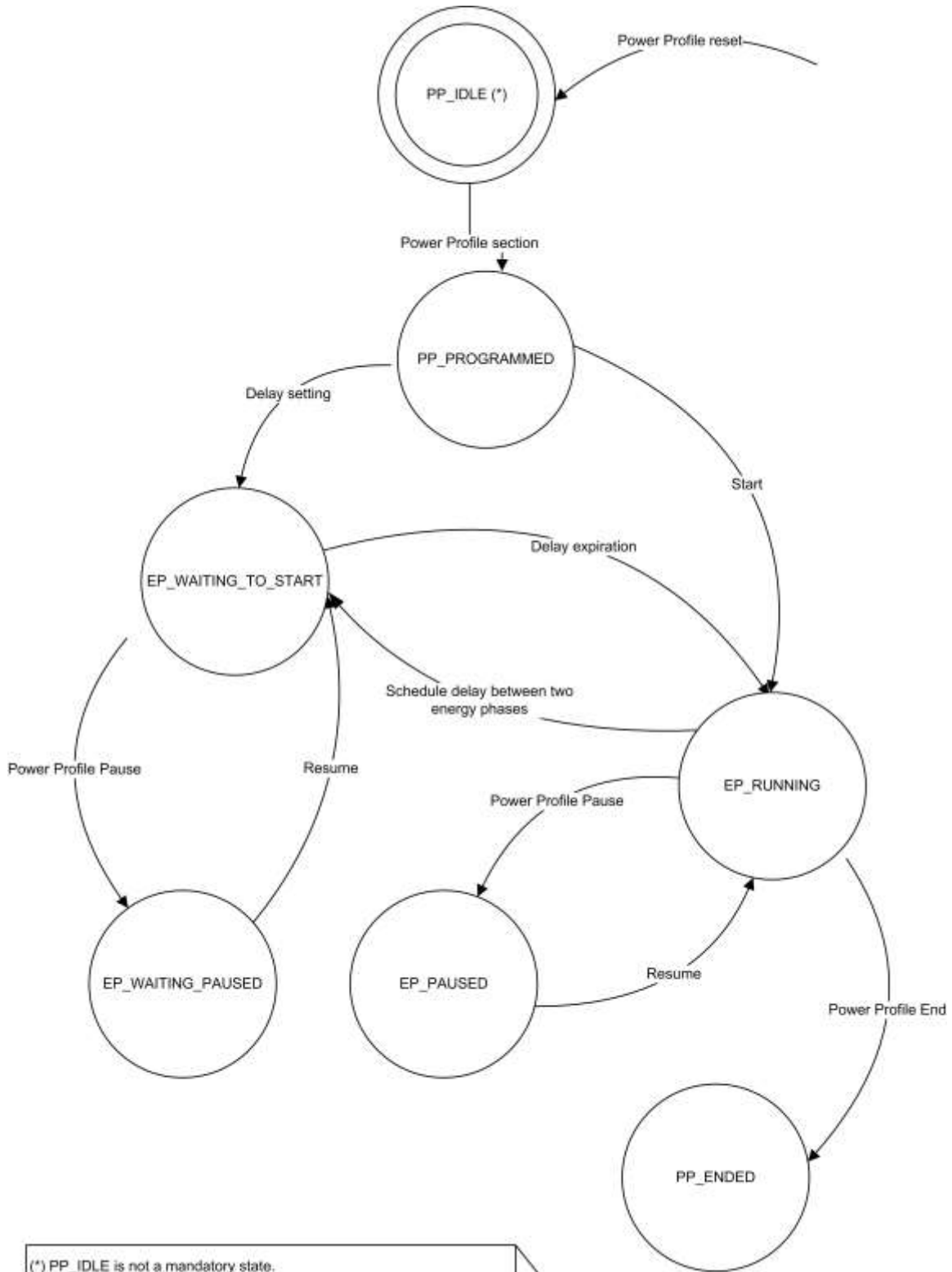
Figure 3-75. Power Profile States



6571

6572

Figure 3-76. Power Profile State Diagram



(\*) PP\_IDLE is not a mandatory state. Some machine could not have this state; in this case the state machine automatically starts with PP\_PROGRAMMED.

6573

6574 **3.17.6.3.2 When Generated**



6575 This command is generated when the command *PowerProfileStateRequest* is received. For more information, see  
6576 Chapter 15, Appliance Management, section 3.

### 6577 **3.17.6.3.3 Effect on Receipt**

6578 On receipt of this command, the originator is notified of the results of its Read Current Power Profiles attempt (i.e.,  
6579 receives the Power Profiles currently running in the server device).

## 6580 **3.17.6.4 GetPowerProfilePrice Command**

6581 The *GetPowerProfilePrice* command is generated by the server (e.g., White Goods) in order to retrieve the cost  
6582 associated to a specific Power Profile. This command has the same payload as the Power Profile Request command.  
6583 For more information, see Chapter 15, Appliance Management, section 3.

### 6584 **3.17.6.4.1 Effect on Receipt**

6585 On receipt of this command, the recipient device SHALL generate a *GetPowerProfilePriceResponse* command. For  
6586 more information, see Chapter 15, Appliance Management, section 3.

## 6587 **3.17.6.5 PowerProfileStateNotification Command**

6588 The *PowerProfileStateNotification* command is generated by the server (e.g., White Goods) in order to update the  
6589 state of the power profile and the current energy phase. It has the same payload as the *PowerProfileStateResponse*  
6590 command but it is an unsolicited command.

### 6591 **3.17.6.5.1 Effect on Receipt**

6592 On receipt of this command, the recipient device will update its information related to the PowerProfile of the device  
6593 (e.g., it will update the forecasts of the durations of the Power Profile's energy phases with the actual data).

## 6594 **3.17.6.6 GetOverallSchedulePrice Command**

6595 The *GetOverallSchedulePrice* command is generated by the server (e.g., White Goods) in order to retrieve the overall  
6596 cost associated to all the Power Profiles scheduled by the scheduler (the device supporting the Power Profile cluster  
6597 client side) for the next 24 hours. This command has no payload.

### 6598 **3.17.6.6.1 Effect on Receipt**

6599 On receipt of this command, the recipient device SHALL generate a *GetOverallSchedulePriceResponse* command.  
6600 For more information, see Chapter 15, Appliance Management, section 3.

## 6601 **3.17.6.7 EnergyPhasesScheduleRequest Command**

6602 The *EnergyPhasesScheduleRequest* Command is generated by the server (e.g., White Goods) in order to retrieve from  
6603 the scheduler (e.g., Home Gateway) the schedule (if available) associated to the specific Power Profile carried in the  
6604 payload. This command has the same payload as the Power Profile Request. For more information, see Chapter 15,  
6605 Appliance Management, section 3.

6606 **3.17.6.7.1 Effect on Receipt**

6607 On receipt of this command, the recipient device SHALL generate an *EnergyPhasesScheduleResponse* command in  
 6608 order to notify the proper scheduling to the server side of the Power Profile cluster (“solicited” schedule). For more  
 6609 information, see Chapter 15, Appliance Management, section 3. If the schedule is accepted by the PowerProfile server  
 6610 side (e.g., the appliance) the PowerProfile SHALL have the state ENERGY\_PHASE\_WAITING\_TO\_START (delay  
 6611 start set for the first energy phase of the power profile). If the device receiving the *EnergyPhasesScheduleResponse*  
 6612 command cannot accept the schedule of the energy phases because the activation delay related to any of carried phases  
 6613 is equal to zero, it SHALL reply with a standard Default response with the error code NOT\_AUTHORIZED (0x7e).

6614 **3.17.6.8 EnergyPhasesScheduleStateResponse Command**

6615 The *EnergyPhasesScheduleStateResponse* command is generated by the server (e.g., White Goods) in order to reply  
 6616 to an *EnergyPhasesScheduleStateRequest* command about the scheduling states that are set in the server side. For  
 6617 more information, see Chapter 15, Appliance Management, section 3. The payload of this command is the same as  
 6618 *EnergyPhasesScheduleNotification*. In case of no scheduled energy phases, the payload shown in Figure 3-77 SHALL  
 6619 be used.

6620 **Figure 3-77. Format of *EnergyPhasesScheduleStateResponse* in Case of No Scheduled Phases**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	PowerProfileID	Num of Scheduled Energy Phases=0x00

6621 **3.17.6.8.1 Effect on Receipt**

6622 On receipt of this command, the recipient device will be notified about the scheduling activity of the server side of the  
 6623 Power Profile Cluster.

6624 Please note that the schedules MAY be set by the scheduling commands listed in this cluster or by the users (e.g.,  
 6625 delay start of an appliance).

6626 **3.17.6.9 EnergyPhasesScheduleStateNotification Command**

6627 The *EnergyPhasesScheduleStateNotification* command is generated by the server (e.g., White Goods) in order to  
 6628 notify (un-solicited command) a client side about the scheduling states that are set in the server side. The payload of  
 6629 this command is the same as *EnergyPhasesScheduleStateResponse*.

6630 **3.17.6.9.1 Effect on Receipt**

6631 On receipt of this command, the recipient devices will be notified about the scheduling activity of the server side of  
 6632 the Power Profile Cluster.

6633 **3.17.6.10 PowerProfileScheduleConstraintsNotification Command**  
6634

6635 The *PowerProfileScheduleConstraintsNotification* command is generated by a device supporting the server side of  
 6636 the Power Profile cluster to notify the client side of this cluster about the imposed constraints and let the scheduler  
 6637 (i.e., the entity supporting the Power Profile cluster client side) to set the proper boundaries for the scheduling.

6638 **3.17.6.10.1 Payload Format**

6639 The *PowerProfileScheduleConstraintsNotification* command payload is reported in Figure 3-78.

6640 **Figure 3-78. Format of the *PowerProfileScheduleConstraintsNotification* Command Frame**

<b>Octets</b>	1	2	2
<b>Data Type</b>	uint8	uint16	uint16
<b>Field Name</b>	PowerProfileID	Start After	Stop Before

6641 **3.17.6.10.1.1 Payload Details**

6642 The payload of the *PowerProfileScheduleConstraintsNotification* command carries the following fields:

- 6643 • The Power Profile ID field specifies which profile (among *TotalProfileNum* total profiles number) the  
6644 constraints are referring to.
- 6645 • The *StartAfter* parameter represents the relative time in minutes (in respect to the time of the reception of this  
6646 command), that limits the start of the Power Profile; it means that the Power Profile SHOULD be scheduled to  
6647 start after a period of time equal to *StartAfter*; if this value is not specified by the device the value SHALL be  
6648 0x0000;
- 6649 • The *StopBefore* parameter represents the relative time in minutes (in respect to the time of the reception of this  
6650 command), that limits the end of the Power Profile; it means that the Power Profile SHOULD be scheduled to  
6651 end before a period of time equal to *StopBefore*; if this value is not specified by the device the value SHALL be  
6652 0xFFFF.

6653 **3.17.6.10.2 When Generated**

6654 This command is generated when the server side of the Power Profile cluster (e.g., a White Goods device), needs to  
6655 notify a change in the constraints of the Power Profile (e.g., the user selected boundaries for the specific behavior of  
6656 the device).

6657 **3.17.6.10.3 Effect on Receipt**

6658 The device that receives the *PowerProfileScheduleConstraintsNotification* command SHALL use the information  
6659 carried in the payload of this command to refine the proper schedule of the specific Power Profile indicated in the  
6660 Power Profile ID field in order to meet the constraints.

6661 **3.17.6.11 PowerProfileScheduleConstraintsResponse  
6662 Command**

6663 The *PowerProfileScheduleConstraintsResponse* command is generated by a device supporting the server side of the  
6664 Power Profile cluster to reply to a client side of this cluster which sent a *PowerProfileScheduleConstraintsRequest*.  
6665 The payload carries the selected constraints to let the scheduler (i.e., the entity supporting the Power Profile client  
6666 cluster) to set the proper boundaries for completing or refining the scheduling.

6667 **3.17.6.11.1 Payload Format**

6668 Same as *PowerProfileScheduleConstraintsNotification* command. For more information, see Chapter 15, Appliance  
6669 Management, section 3.

6670 **3.17.6.11.2 When Generated**

6671 This command is generated as a reply to the Power Profile Schedule Constraints Request.

6672 **3.17.6.11.3 Effect on Receipt**6673 The device that receives the Power Profile Schedule ConstraintsResponse command SHALL use the information  
6674 carried in the payload of this command to refine the proper schedule of the specific Power Profile indicated in the  
6675 Power ProfileID field.6676 **3.17.6.12 GetPowerProfilePriceExtended Command**6677 The *GetPowerProfilePriceExtended* command is generated by the server (e.g., White Goods) in order to retrieve the  
6678 cost associated to a specific Power Profile considering specific conditions described in the option field (e.g., a specific  
6679 time).6680 **3.17.6.12.1 Payload Format**6681 The *GetPowerProfilePriceExtended* command payload SHALL be formatted as illustrated in Figure 3-79.6682 **Figure 3-79. Format of the *GetPowerProfilePriceExtended* Command Payload**

<b>Octets</b>	1	1	0/2
<b>Data Type</b>	map8	uint8	uint16
<b>Field Name</b>	Options	PowerProfileID	PowerProfileStartTime

6683

6684 **Table 3-141. Options Field**

<b>Bit</b>	<b>Description</b>
0	Bit0=1 : PowerProfileStartTime Field Present
1	Bit1=0 : provide an estimation of the price considering the power profile with contiguous energy phases Bit1=1 : provide an estimation of the price considering the power profile as scheduled (i.e., taking in account delays between Energy phases set by the EMS)

6685

6686 **Options**6687 The Options field represents the type of request of extended price is requested to the client side of the power profile  
6688 cluster (e.g., to an energy management system).6689 **PowerProfileStartTime**6690 The PowerProfileStartTime field represents the relative time (expressed in relative encoding in respect to the current  
6691 time) when the overall Power Profile can potentially start. The unit is the minute.

6692 **3.17.6.12.2 Effect on Receipt**

6693 On receipt of this command, the recipient device SHALL generate a *GetPowerProfilePriceExtendedResponse*  
6694 command. For more information, see Chapter 15, Appliance Management, section 3.

6695 **3.17.7 Client Attributes**

6696 The client has no cluster specific attributes.

6697 **3.17.8 Client Commands Received**

6698 Description is in server side commands generated (sent) description.

6699 **3.17.9 Client Commands Generated**

6700 Description is in server side commands received description.

6701 **3.17.10 Example of Device Interactions Using the Power**  
6702 **Profile (Informative Section)**

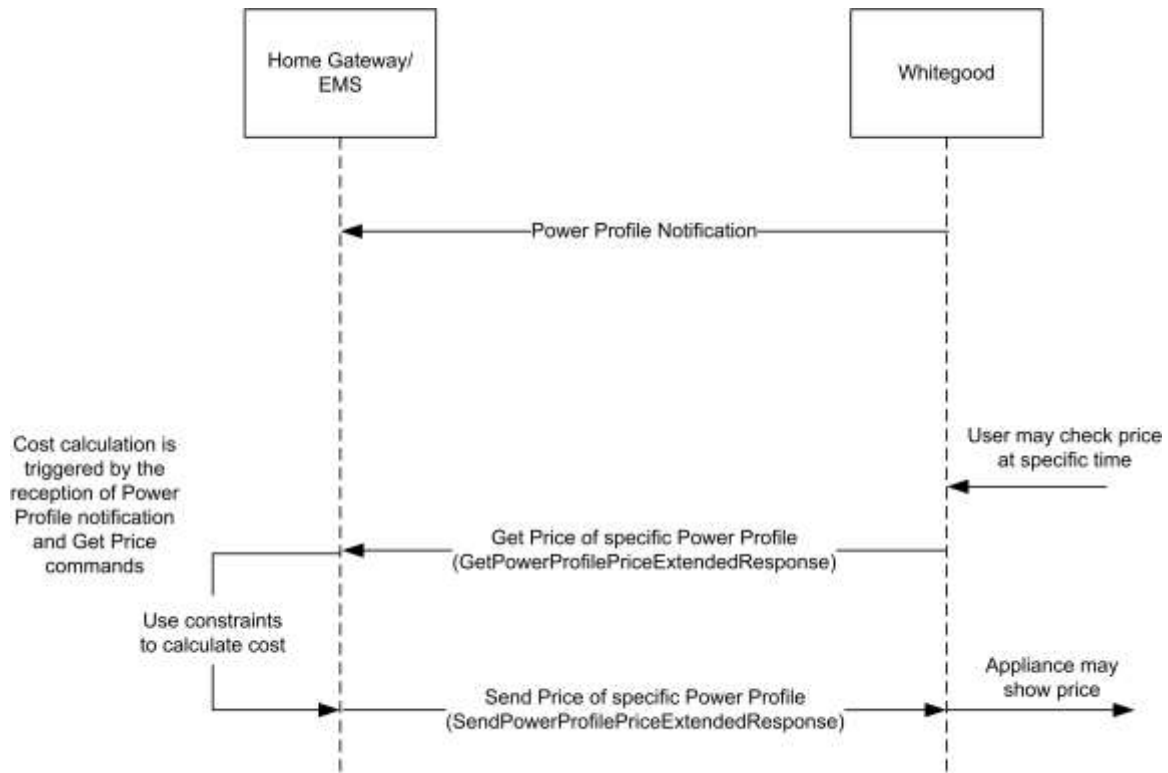
6703 **3.17.10.1 Price Information Retrieved by the White Goods**

6704 The price of a specific appliance program is estimated by the Home gateway/EMS, calculated using the Power Profile  
6705 forecast provided by the appliance and the *PowerProfileStartTime* contained in the *GetPowerProfilePriceExtended*  
6706 which indicates when the appliance program will start. How the Home Gateway/EMS retrieves from the utility the  
6707 information related to tariff schemes and price changes over time is out of scope of this specification.

6708 The appliance MAY then show to the user on the display the price associated to a specific cycle set (e.g., a washing  
6709 machine program “Cotton 90 °C” will cost you “1.15€”).

6710

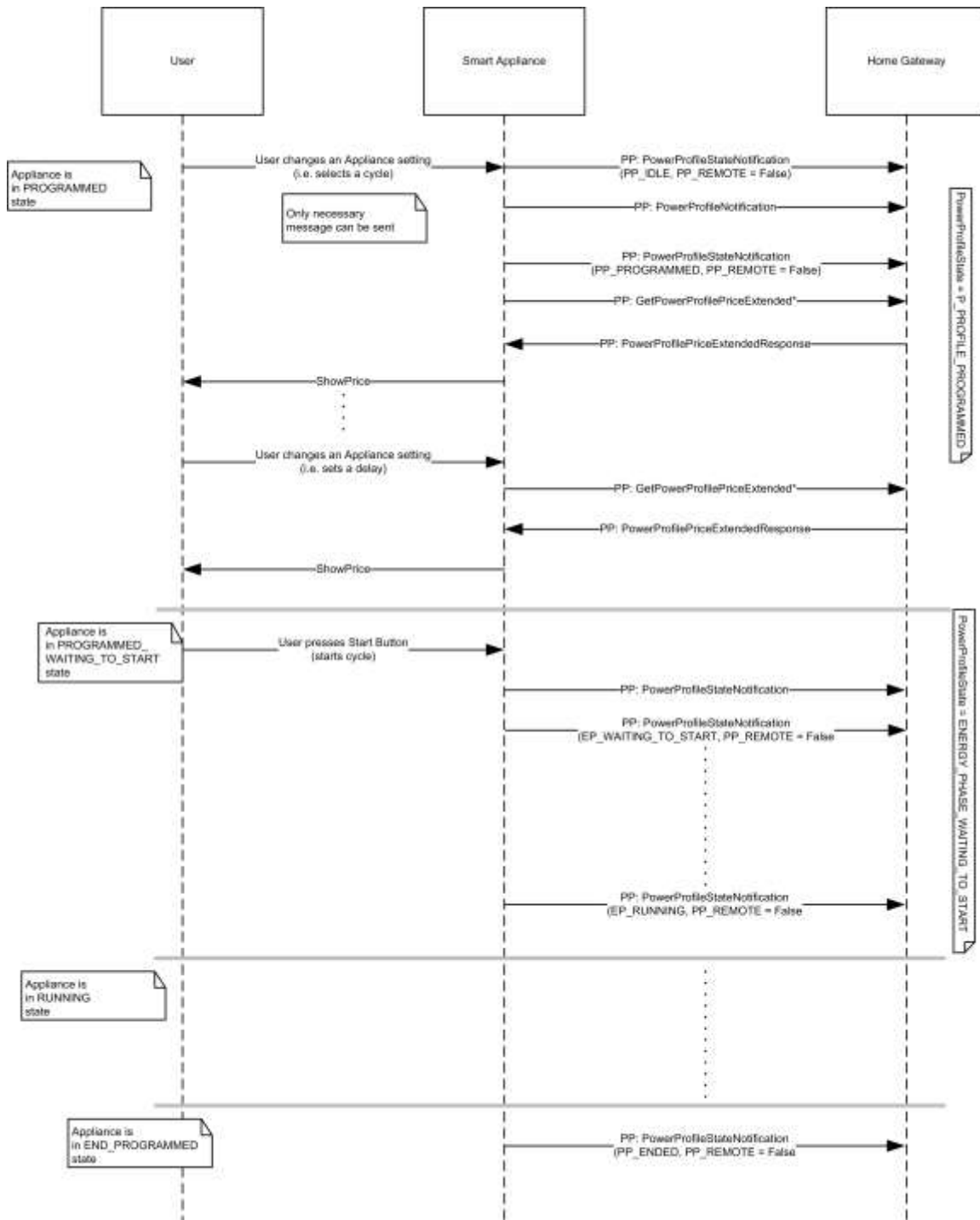
Figure 3-80. Visualization of Price Associated to a Power Profile



6711

6712 **3.17.10.2 Interaction with Power Profile Cluster When**  
 6713 **Appliance Is Not Remotely Controllable**

6714 **Figure 3-81. Energy Remote Disabled: Example of Sequence Diagram with User Interaction**

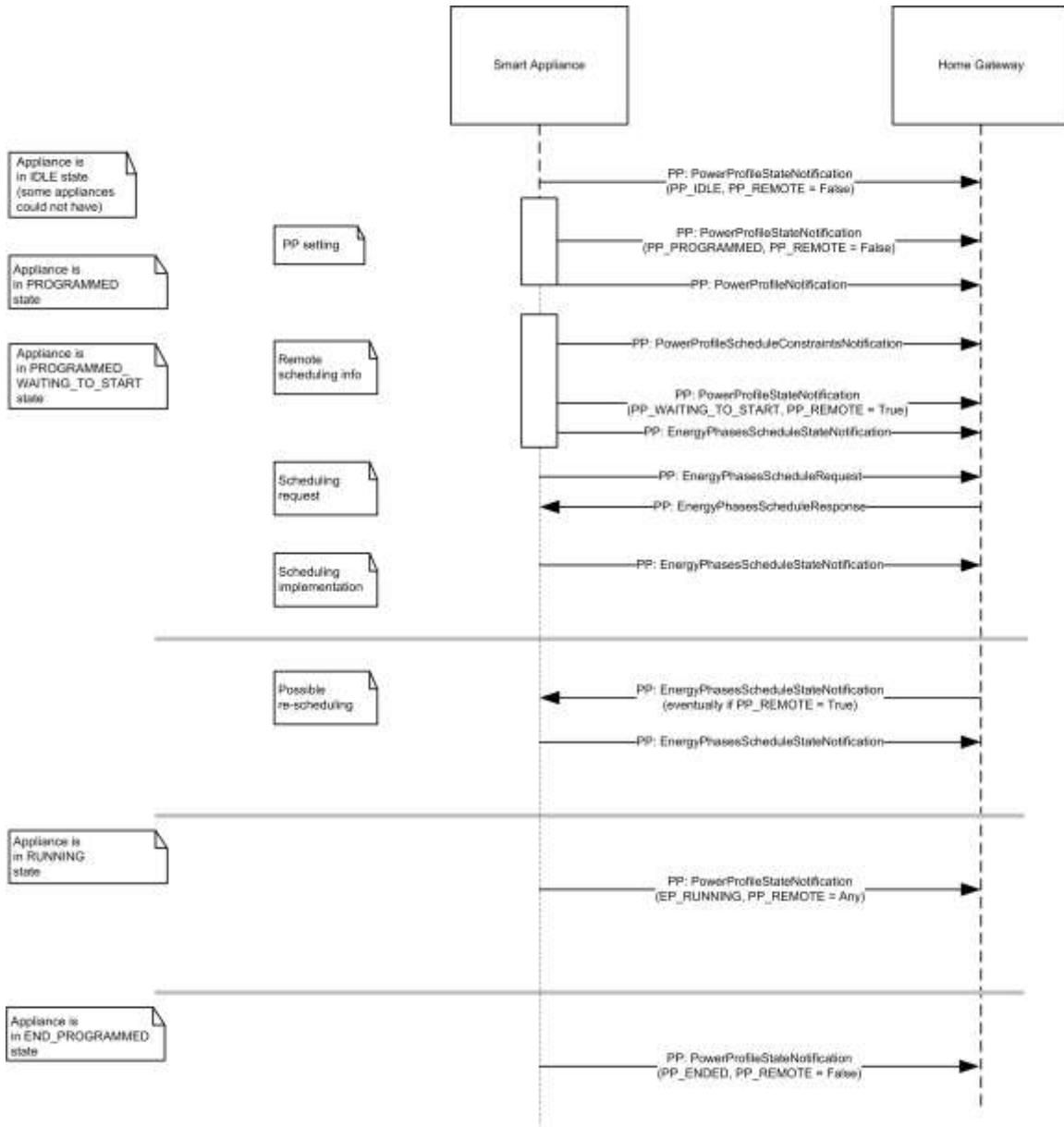


\*GetPowerProfilePriceExtended payload includes delay time to start

6715  
 6716

6717 **3.17.10.3 Interaction with Power Profile Cluster When Appliance Is Remotely Controllable (Scheduling of Appliance)**  
6718  
6719

6720 **Figure 3-82. Energy Remote Enabled: Example of Sequence Diagram with User Interaction**



6721 \*GetPowerProfilePriceExtended can be generated any time by SA if a PP is active

6722 **3.18 Meter Identification**

6723 **3.18.1 Overview**

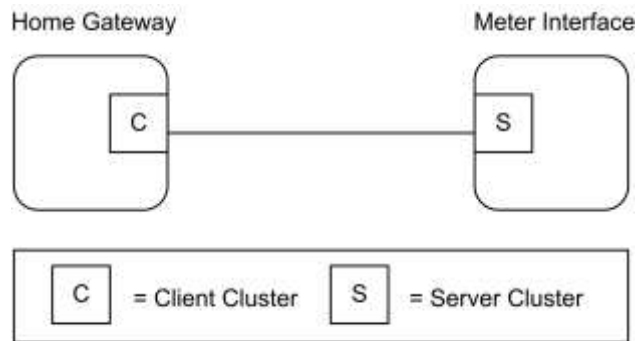


6724 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 6725 etc.

6726 This cluster provides attributes and commands for determining advanced information about utility metering device,  
 6727 as shown in Figure 3-83.

6728 **Note:** Where a physical node supports multiple endpoints it will often be the case that many of these settings will apply  
 6729 to the whole node, that is they are the same for every endpoint on the device. In such cases they can be implemented  
 6730 once for the node, and mapped to each endpoint.

6731 **Figure 3-83. Typical Usage of the Meter Identification Cluster**



Note: Device names are examples for illustration purposes only

6732

6733 **3.18.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

6734 **3.18.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MTRID	Type 2 (server to client)

6735 **3.18.1.3 Cluster Identifiers**

Identifier	Name
0x0b01	Meter Identification

6736 **3.18.2 Server**

6737 **3.18.2.1 Meter Identification Attribute Set**

6738 The Meter Identification server cluster contains the attributes summarized in Table 3-142.

6739

**Table 3-142. Attributes of the Meter Identification Server Cluster**

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	<i>CompanyName</i>	string	0 to 16 Octets	R	-	M
0x0001	<i>MeterTypeID</i>	uint16	0x0000 to 0xffff	R	-	M
0x0004	<i>DataQualityID</i>	uint16	0x0000 to 0xffff	R	-	M
0x0005	<i>CustomerName</i>	string	0 to 16 Octets	RW	-	O
0x0006	<i>Model</i>	octstr	0 to 16 Octets	R	-	O
0x0007	<i>PartNumber</i>	octstr	0 to 16 Octets	R	-	O
0x0008	<i>ProductRevision</i>	octstr	0 to 6 Octets	R	-	O
0x000A	<i>SoftwareRevision</i>	octstr	0 to 6 Octets	R	-	O
0x000B	<i>UtilityName</i>	string	0 to 16 Octets	R	-	O
0x000C	<i>POD</i>	string	0 to 16 Octets	R	-	M
0x000D	<i>AvailablePower</i>	int24	0x000000 to 0xfffffff	R	-	M
0x000E	<i>PowerThreshold</i>	int24	0x000000 to 0xfffffff	R	-	M

6740 **3.18.2.1.1 CompanyName Attribute**

6741 *CompanyName* is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length)  
6742 encoded in the UTF-8 format. Company Name defines the meter manufacturer name, decided by manufacturer.

6743 **3.18.2.1.2 MeterTypeID Attribute**

6744 *MeterTypeID* defines the Meter installation features, decided by manufacturer. Table 3-143 provides Meter Type IDs  
6745 field content.

6746

**Table 3-143. Meter Type IDs**

Device	Meter Type ID
Utility Primary Meter	0x0000
Utility Production Meter	0x0001
Utility Secondary Meter	0x0002
Private Primary Meter	0x0100
Private Production Meter	0x0101
Private Secondary Meters	0x0102
Generic Meter	0x0110

6747

6748 **3.18.2.1.3 DataQualityID Attribute**

6749 *DataQualityID* defines the Meter Simple Metering information certification type, decided by manufacturer.

6750 Table 3-144 provides Data Quality IDs field content.

6751 **Table 3-144. Data Quality IDs**

Device	Meter Type ID
All Data Certified	0x0000
Only Instantaneous Power not Certified	0x0001
Only Cumulated Consumption not Certified	0x0002
Not Certified data	0x0003

6752

6753 **3.18.2.1.4 CustomerName Attribute**

6754 *CustomerName* is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the ASCII format.

6756 **3.18.2.1.5 Model Attribute**

6757 *Model* is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *Model* defines the meter model name, decided by manufacturer.

6759 **3.18.2.1.6 PartNumber Attribute**

6760 *PartNumber* is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the UTF-8 format. *PartNumber* defines the meter part number, decided by manufacturer.

6762 **3.18.2.1.7 ProductRevision Attribute**

6763 *ProductRevision* is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *ProductRevision* defines the meter revision code, decided by manufacturer.

6765 **3.18.2.1.8 SoftwareRevision Attribute**

6766 *SoftwareRevision* is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length) encoded in the UTF-8 format. *SoftwareRevision* defines the meter software revision code, decided by manufacturer.

6768 **3.18.2.1.9 UtilityName Attribute**

6769 *UtilityName* is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length) encoded in the ASCII format.

6770

6771 **3.18.2.1.10 POD Attribute**

6772 *POD (Point of Delivery)* is a ZCL Character String field capable of storing up to 16 character string (the first Octet  
6773 indicates length) encoded in the ASCII format. *POD* is the unique identification ID of the premise connection point.  
6774 It is also a contractual information known by the clients and indicated in the bill.

6775 **3.18.2.1.11 AvailablePower Attribute**

6776 *AvailablePower* represents the *InstantaneousDemand* that can be distributed to the customer (e.g., 3.3KW power)  
6777 without any risk of overload. The *Available Power* SHALL use the same formatting conventions as the one used in  
6778 the simple metering cluster formatting attribute set for the *InstantaneousDemand* attribute, i.e., the *UnitOfMeasure*  
6779 and *DemandFormatting*.

6780 **3.18.2.1.12 PowerThreshold Attribute**

6781 *PowerThreshold* represents a threshold of *InstantaneousDemand* distributed to the customer (e.g., 4.191KW) that will  
6782 lead to an imminent risk of overload. The *PowerThreshold* SHALL use the same formatting conventions as the one  
6783 used in the *AvailablePower* attributes and therefore in the simple metering cluster formatting attribute set for the  
6784 *InstantaneousDemand* attribute, i.e., the *UnitOfMeasure* and *DemandFormatting*.

6785 **3.18.2.1.13 Commands Received**

6786 No cluster-specific commands are received or generated by the server.

6787 **3.18.3 Client**

6788 The client has no dependencies and no cluster specific attributes. The client does not receive or generate any cluster  
6789 specific commands.

6790 **3.19 Level Control for Lighting**6791 **3.19.1 Overview**

6792 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
6793 etc.

6794 This cluster provides an interface for controlling the level of a light source.

6795 All requirements and dependencies are derived from the base cluster. Additional or extended requirements and  
6796 dependencies are listed in this cluster specification.

6797

6798 NOTE: This cluster specification is derived from the Level cluster specification (generic level control also defined in  
6799 this document). This cluster specifies further requirements for the lighting application. Please see section 3.10 for the  
6800 generic Level cluster specification.

6801 **3.19.1.1 Revision History**

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added
2	added <i>Options</i> attribute, state change table; ZLO 1.0; Derived from base Level (no change)

CCB 2085 1775 2281 2147
-------------------------

6802 **3.19.1.2 Classification**

Hierarchy	Base	Role	PICS Code	Primary Transaction
Derived	Level (3.10)	Application	LC	Type 1 (client to server)

6803 **3.19.1.3 Cluster Identifiers**

Identifier	Name
0x0008	Level Control for Lighting

6804 **3.19.2 Server**

6805 **3.19.2.1 Dependencies**

6806 Please see examples of state changes with regards to the On/Off server cluster in section 3.19.4.

6807 **3.19.2.2 Attributes**

6808 The Level Control for Lighting server cluster supports the base attributes below. See the base cluster for details.  
 6809 Additional requirements are defined here.

6810

6811 **Table 3-145. Attributes of the Level Control for Lighting server cluster**

Name	Range	Default	M/O
<i>CurrentLevel</i>	0x01 to 0xfe	0xff	M
<i>RemainingTime</i>	<i>base</i> <sup>43</sup>	<i>base</i>	O
<i>CurrentFrequency</i>	<i>base</i>	<i>base</i>	O
<i>MinFrequency</i>	<i>base</i>	<i>base</i>	M: <i>CurrentFrequency</i> <sup>44</sup>
<i>MaxFrequency</i>	<i>base</i>	<i>base</i>	M: <i>CurrentFrequency</i>
<i>OnOffTransitionTime</i>	<i>base</i>	<i>base</i>	O
<i>OnLevel</i>	<i>base</i>	<i>base</i>	O
<i>OnTransitionTime</i>	<i>base</i>	<i>base</i>	O
<i>OffTransitionTime</i>	<i>base</i>	<i>base</i>	O

<sup>43</sup> see base cluster for definition

<sup>44</sup> Mandatory if *CurrentFrequency* supported

Name	Range	Default	M/O
<i>DefaultMoveRate</i>	<i>base</i>	<i>base</i>	O
<i>Options</i>	<i>base</i>	<i>base</i>	M
<i>StartUpCurrentLevel</i>	<i>base</i>	<i>base</i>	O

6812 **3.19.2.2.1 CurrentLevel Attribute for Lighting**

6813 A value of 0x00 SHALL not be used.

6814 A value of 0x01 SHALL indicate the minimum level that can be attained on a device.

6815 A value of 0xfe SHALL indicate the maximum level that can be attained on a device.

6816 A value of 0xff SHALL represent an undefined value.

6817 All other values are application specific gradations from the minimum to the maximum level.

6818 **3.19.2.2.2 CurrentFrequency Attribute**

6819 The *CurrentFrequency* attribute represents the frequency in 10Hz (hertz) increments up to 655.34 kHz. A value of 0  
6820 is unknown.

6821 **3.19.2.2.3 Options Attribute for Lighting**

6822 Below describes the lighting specific bits of the *Options* attribute. All other bits are defined, or reserved by the base  
6823 cluster.

6824 **Table 3-146. Options Attribute**

Bit	Name	Values & Summary
1	CoupleColorTempToLevel (See Color Control cluster)	0 - Do not couple changes to the <i>CurrentLevel</i> attribute with the color temperature. 1 - Couple changes to the <i>CurrentLevel</i> attribute with the color temperature.

6825 **3.19.2.3 Commands Received**

6826 The command IDs for the cluster are listed below:

6827 **Table 3-147. Commands for the Pulse Width Modulation cluster**

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M

ID	Description	M/O
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency	M:CurrentFrequency

6828 **3.19.2.4 Commands Generated**

6829 There are no commands generated by the server.

6830 **3.19.3 Client**

6831 The client has no cluster specific attributes. The client generates the cluster specific commands received by the server,  
 6832 as required by the application. No cluster specific commands are received by the client.

6833 **3.19.4 State Change Table for Lighting**

6834 Below is a table of examples of state changes when Level Control for Lighting and On/Off clusters are on the same  
 6835 endpoint.

6836 *EiO - ExecuteIfOff* field in the *Options* attribute

6837 *OnOff* – attribute value of On/Off cluster 0=Off, 1=On

6838 *MIN - MinLevel*

6839 *MAX - MaxLevel*

6840 *MID* – midpoint between *MinLevel* and *MaxLevel*

6841 **Table 3-148. Lighting Device State Change**

<i>Current Level</i>	<i>EiO</i>	<i>OnOff</i>	<i>Physical Device</i>	<b>Command</b> ← Before After →	<i>Current Level</i>	<i>OnOff</i>	<i>Physical Device</i>	<b>Device Output Result</b>
<i>any</i>	0	0	Off	Move to level <i>MID</i> over 2 sec	<i>same</i>	0	Off	stays off
<i>any</i>	0	0	Off	Move with On/Off to level <i>MID</i> over 2 sec	<i>MID</i>	1	On (mid-point brightness)	turns on and output level adjusts or stays at half

<i>Current Level</i>	<i>EiO</i>	<i>OnOff</i>	<b>Physical Device</b>	<b>Command</b> ← Before After →	<i>Current Level</i>	<i>OnOff</i>	<b>Physical Device</b>	<b>Device Output Result</b>
<i>any</i>	1	0	Off	Move to level <i>MID</i> over 2 sec	<i>MID</i>	0	Off	stays off
<i>any</i>	1	0	Off	Move with On/Off to level <i>MID</i> over 2 sec	<i>MID</i>	1	On	turns on and output level adjusts to or stays at half
<i>any</i>	1	0	Off	Move rate = up 64 per second	<i>MAX</i>	0	Off	stays off
<i>any</i>	1	0	Off	Move with On/Off rate = up 64 per second	<i>MAX</i>	1	On	turn on and output level adjusts to or stays at full
<i>any</i>	1	0	Off	Move (with On/Off) rate = down 64 per second	<i>MIN</i>	0	Off	stays off
<i>any</i>	<i>any</i>	1	On (any brightness)	Move (with On/Off) to level <i>MID</i> over 2 sec	<i>MID</i>	1	On (mid-point brightness)	output level adjusts to or stays at half
<i>any</i>	<i>any</i>	1	On (any brightness)	Move (with On/Off) rate = up 64 per second	<i>MAX</i>	1	On (full brightness)	output level adjusts to or stays at full
<i>any</i>	<i>any</i>	1	On (any brightness)	Move rate = down 64 per second	<i>MIN</i>	0	On (at minimum brightness)	output level adjusts to minimum
<i>any</i>	<i>any</i>	1	On (any brightness)	Move with On/Off rate = down 64 per second	<i>MIN</i>	0	Off	output level adjusts to off

6842



## 6843 3.20 Pulse Width Modulation

### 6844 3.20.1 Overview

6845 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 6846 etc.

6847 This cluster provides an interface for controlling the Pulse Width Modulation (PWM) characteristics of a device.  
 6848 Typical applications include heating element and fan control (10-100Hz), DC electric motors and power efficient LED  
 6849 control (5-10kHz), and switching power supplies (>20kHz). For the purposes of PWM, the value of level is effectively  
 6850 a duty cycle. The frequency and level (duty cycle) values are reportable and may be configured for reporting.

#### 6851 3.20.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

#### 6852 3.20.1.2 Classification

Hierarchy	Base	Role	PICS Code	Primary Transaction
Derived	Level (3.10)	Application	PWM	Type 1 (client to server)

#### 6853 3.20.1.3 Cluster Identifiers

Identifier	Name
0x001c	Pulse Width Modulation

### 6854 3.20.2 Server

6855 The server requirements and dependencies are derived from the base cluster. Additional requirements are listed in this  
 6856 section below.

#### 6857 3.20.2.1 Attributes

6858 The Pulse Width Modulation server cluster supports the base attributes below. See the base cluster for details.  
 6859 Additional requirements are defined here.

6860 **Table 3-149. Attributes of the Pulse Width Modulation server cluster**

Name	Range	Default	M/O
<i>CurrentLevel</i>	<i>MinLevel</i> to <i>MaxLevel</i>	0xff	M
<i>MinLevel</i>	0 to <i>MaxLevel</i>	0	M
<i>MaxLevel</i>	<i>MinLevel</i> to 100	100	M
<i>CurrentFrequency</i>	<i>MinFrequency</i> to <i>MaxFrequency</i>	0	M

Name	Range	Default	M/O
<i>MinFrequency</i>	0 to <i>MaxFrequency</i>	0	M
<i>MaxFrequency</i>	<i>MinFrequency</i> to 0xffff	0	M

### 6861 3.20.2.1.1 CurrentLevel Attribute

6862 The *CurrentLevel* attribute represents a duty cycle as a percentage. A value of 0xff is unknown.

### 6863 3.20.2.1.2 CurrentFrequency Attribute

6864 The *CurrentFrequency* attribute represents the frequency in 10Hz (hertz) increments up to 655.34 kHz. A value of 0  
6865 is unknown.

## 6866 3.20.2.2 Commands Received

6867 The command IDs for the cluster are listed below:

6868 **Table 3-150. Commands for the Pulse Width Modulation cluster**

ID	Description	M/O
0x00	Move to Level	M
0x01	Move	M
0x02	Step	M
0x03	Stop	M
0x04	Move to Level (with On/Off)	M
0x05	Move (with On/Off)	M
0x06	Step (with On/Off)	M
0x07	Stop	M
0x08	Move to Closest Frequency	M

### 6869 3.20.2.3 Commands Generated

6870 There are no commands generated by the server.

## 6871 3.20.3 Client

6872 The client has no cluster specific attributes. The client generates the cluster specific commands received by the server,  
6873 as required by the application. No cluster specific commands are received by the client.

6874

## CHAPTER 4 MEASUREMENT AND SENSING

6875  
6876  
6877  
6878

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

6879

### 4.1 General Description

6880

#### 4.1.1 Introduction

6881  
6882

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

6883

#### 4.1.2 Cluster List

6884

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

6885

##### 4.1.2.1 Illuminance Measurement and Level Sensing

6886

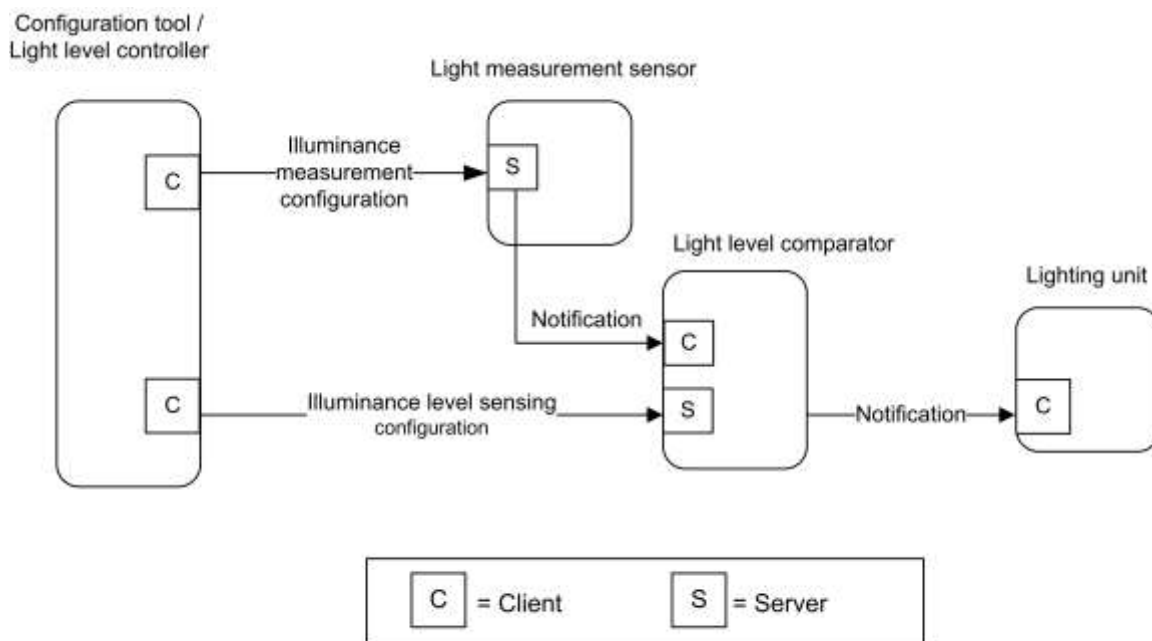
Table 4-1. Illuminance Measurement and Level Sensing Clusters

Cluster ID	Cluster Name	Description
0x0400	Illuminance Measurement	Attributes and commands for configuring the measurement of illuminance, and reporting illuminance measurements
0x0401	Illuminance Level Sensing	Attributes and commands for configuring the sensing of illuminance levels, and reporting whether illuminance is above, below, or on target

6887

6888

Figure 4-1. Typical Usage of Illuminance Measurement and Level Sensing Clusters



6889

Note: Device names are examples for illustration purposes only

6890

## 4.1.2.2 Temperature, Pressure and Flow Measurement

6891

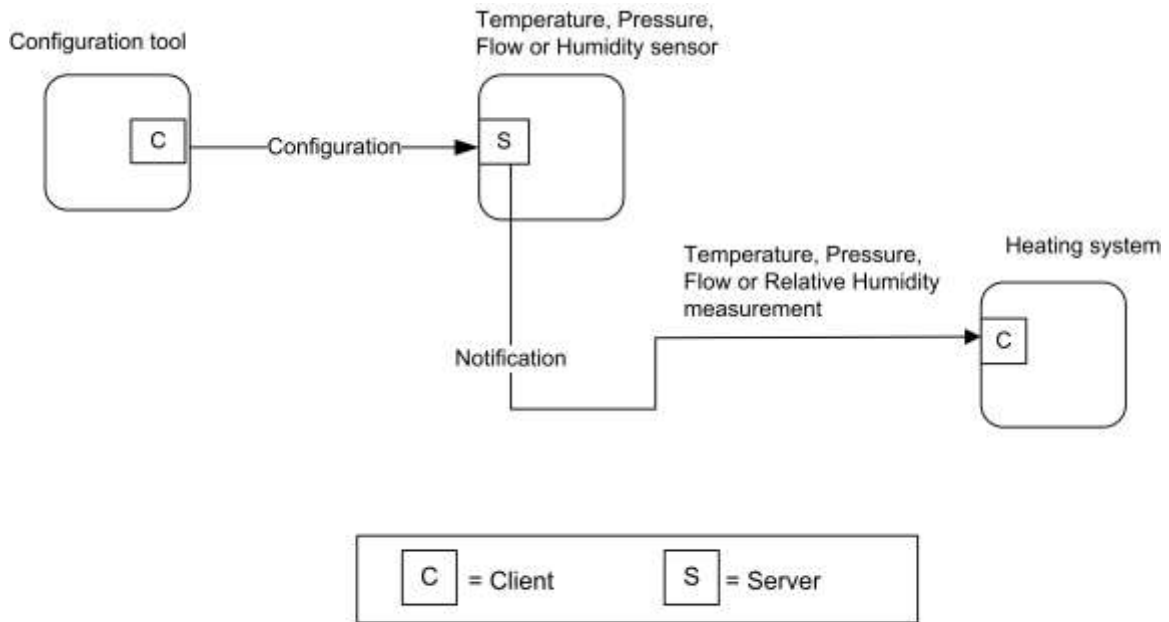
Table 4-2. Pressure and Flow Measurement Clusters

ID	Cluster Name	Description
0x0402	Temperature Measurement	Attributes and commands for configuring the measurement of temperature, and reporting temperature measurements
0x0403	Pressure Measurement	Attributes and commands for configuring the measurement of pressure, and reporting pressure measurements
0x0404	Flow Measurement	Attributes and commands for configuring the measurement of flow, and reporting flow rates
0x0405	Relative Humidity Measurement	Attributes and commands for configuring the measurement of relative humidity, and reporting relative humidity measurements

6892

6893

**Figure 4-2. Typical Usage of Temperature, Pressure and Flow Measurement Clusters**



6894

*Note: Device names are examples for illustration purposes only*

### 6895 4.1.2.3 Occupancy Sensing

6896

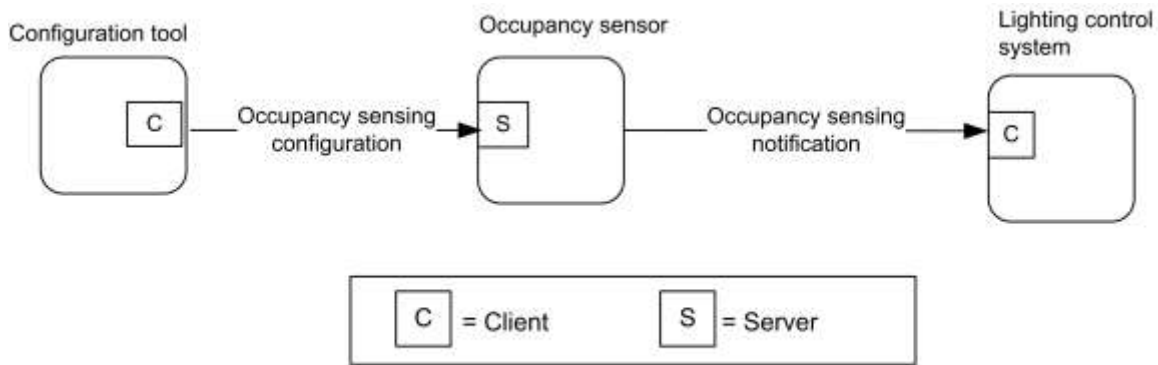
**Table 4-3. Occupancy Sensing Clusters**

ID	Cluster Name	Description
0x0406	Occupancy Sensing	Attributes and commands for configuring occupancy sensing, and reporting occupancy status

6897

6898

Figure 4-3. Typical Usage of Occupancy Sensing Cluster



6899

Note: Device names are examples for illustration purposes only

#### 6900 4.1.2.4 Electrical Measurement

6901 Table 4-4. Electrical Measurement Clusters

Cluster ID	Cluster Name	Description
0x0b04	Electrical Measurement	Attributes and commands for measuring electrical usage

### 6902 4.1.3 Measured Value<sup>45</sup>

#### 6903 4.1.3.1 Range

6904 For any measurement cluster with *MeasuredValue*, *MinMeasuredValue* and *MaxMeasuredValue* attributes, the  
6905 following SHALL be always be true:

6906

- 6907 • If both are defined, then *MaxMeasuredValue* SHALL be greater than *MinMeasuredValue*.
- 6908 • If *MaxMeasuredValue* is known, then *MeasuredValue* SHALL be less than or equal to *MaxMeasuredValue*.
- 6909 • If *MinMeasuredValue* is known, then *MeasuredValue* SHALL be greater than or equal to *MinMeasuredValue*.

#### 6910 4.1.3.2 Tolerance

6911 For any measurement cluster with a *MeasuredValue* and *Tolerance* attribute, the following SHALL always be true:

6912 The *Tolerance* attribute SHALL indicate the magnitude of the possible error that is associated with  
6913 *MeasuredValue*, using the same units and resolution. The true value SHALL be in the range (*MeasuredValue*  
6914 – *Tolerance*) to (*MeasuredValue* + *Tolerance*).

6915 If known, the true value SHALL never be outside the possible physical range. Some examples:

- 6916 • a temperature SHALL NOT be below absolute zero
- 6917 • a concentration SHALL NOT be negative

<sup>45</sup> NFR Quality of Goods

6918 **4.2 Illuminance Measurement**

6919 **4.2.1 Overview**

6920 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 6921 etc.

6922 The server cluster provides an interface to illuminance measurement functionality, including configuration and  
 6923 provision of notifications of illuminance measurements.

6924 **4.2.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCB 2048 2049 2050
2	CCB 2167

6925 **4.2.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ILL	Type 2 (server to client)

6926 **4.2.1.3 Cluster Identifiers**

Identifier	Name
0x0400	Illuminance Measurement

6927 **4.2.2 Server**

6928 **4.2.2.1 Attributes**

6929 The Illuminance Measurement attributes summarized in Table 4-5.

6930 **Table 4-5. Illuminance Measurement Attributes**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	uint16	0x0000 to 0xffff	RP	0x0000	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0001 – 0xfffd	R <sup>46</sup>	ms	M
0x0002	<i>MaxMeasuredValue</i>	uint16	2 to 0xfffe	R	ms	M
0x0003	<i>Tolerance</i>	uint16	0x0000 – 0x0800	R	ms	O
0x0004	<i>LightSensorType</i>	enum8	0x00 – 0xff	R	0xff	O

<sup>46</sup> CCB 2167 typo: *MeasuredValue* should be reportable (not *MinMeasuredValue*)

6931 **4.2.2.1.1 MeasuredValue Attribute**

6932 *MeasuredValue* represents the Illuminance in Lux (symbol lx) as follows:

6933  $MeasuredValue = 10,000 \times \log_{10} \text{Illuminance} + 1$

6934 Where  $1 \text{ lx} \leq \text{Illuminance} \leq 3.576 \text{ Mlx}$ , corresponding to a *MeasuredValue* in the range 1 to 0xffff.

6935 The *MeasuredValue* attribute can take the following values.

- 6936 • 0x0000 indicates a value of Illuminance that is too low to be measured.
- 6937 •  $MinMeasuredValue \leq MeasuredValue \leq MaxMeasuredValue$  under normal circumstances.
- 6938 • 0xffff indicates that the Illuminance measurement is invalid.

6939 *MeasuredValue* is updated continuously as new measurements are made.

6940 **4.2.2.1.2 MinMeasuredValue Attribute**

6941 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of  
6942 0xffff indicates that this attribute is not defined. See 4.1.3 for more details.

6943 **4.2.2.1.3 MaxMeasuredValue Attribute**

6944 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of  
6945 0xffff indicates that this attribute is not defined. See 4.1.3 for more details.

6946 **4.2.2.1.4 Tolerance Attribute**

6947 See 4.1.3.

6948 **4.2.2.1.5 LightSensorType Attribute**

6949 The *LightSensorType* attribute specifies the electronic type of the light sensor. This attribute shall be set to one of the  
6950 non-reserved values listed in Table 4-6.

6951 **Table 4-6. Values of the *LightSensorType* Attribute**

Attribute Value	Description
0x00	Photodiode
0x01	CMOS
0x40 – 0xfe	Reserved for manufacturer specific light sensor types
0xff	Unknown

6952 **4.2.2.2 Commands**

6953 No cluster specific commands are generated or received by the server cluster.



6954 **4.2.2.3 Attribute Reporting**

6955 This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and  
 6956 maximum reporting intervals and reportable change settings described in the ZCL Foundation specification (see 2.4.7).  
 6957 The following attributes shall be reported:

6958 *MeasuredValue*

6959 **4.2.3 Client**

6960 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

6961 **4.3 Illuminance Level Sensing**

---

6962 **4.3.1 Overview**

6963 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 6964 etc.

6965 The server cluster provides an interface to illuminance level sensing functionality, including configuration and  
 6966 provision of notifications of whether the illuminance is within, above or below a target band.

6967 **4.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

6968 **4.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	ILLVL	Type 2 (server to client)

6969 **4.3.1.3 Cluster Identifiers**

Identifier	Name
0x0401	Illuminance Level Sensing

6970 **4.3.2 Server**

6971 **4.3.2.1 Attributes**

6972 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 6973 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 6974 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 6975 are listed in Table 4-7.

6976

**Table 4-7. Illuminance Level Sensing Attribute Sets**

Attribute Set Identifier	Description
0x000	Illuminance Level Sensing Information
0x001	Illuminance Level Sensing Settings

6977

### 4.3.2.2 Illuminance Level Sensing Information Attribute Set

6978

The light sensor configuration attribute set contains the attributes summarized in Table 4-8.

6979

**Table 4-8. Illuminance Level Sensing Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>LevelStatus</i>	enum8	0x00 – 0xfe	RP	-	M
0x0001	<i>LightSensorType</i>	enum8	0x00 – 0xfe	R	-	O

6980

#### 4.3.2.2.1 *LevelStatus* Attribute

6981

The *LevelStatus* attribute indicates whether the measured illuminance is above, below, or within a band around *IlluminanceTargetLevel* (see 4.3.2.3.1). It may have any non-reserved value shown in Table 4-9.

6982

6983

**Table 4-9. Values of the *LevelStatus* Attribute**

Attribute Value	Description
0x00	Illuminance on target
0x01	Illuminance below target
0x02	Illuminance above target

6984

#### 4.3.2.2.2 *LightSensorType* Attribute

6985

The *LightSensorType* attribute specifies the electronic type of the light sensor. This attribute shall be set to one of the non-reserved values listed in Table 4-10.

6986

6987

**Table 4-10. Values of the *LightSensorType* Attribute**

Attribute Value	Description
0x00	Photodiode
0x01	CMOS
0x40 – 0xfe	Reserved for manufacturer specific light sensor types
0xff	Unknown

6988 **4.3.2.3 Illuminance Level Sensing Settings Attribute Set**

6989 The light sensor configuration attribute set contains the attributes summarized in Table 4-11 Illuminance Level Sensing  
 6990 Settings Attribute Set.

6991 **Table 4-11. Illuminance Level Sensing Settings Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0010	<i>IlluminanceTargetLevel</i>	uint16	0x0000 – 0xffff	RW	-	M

6992 **4.3.2.3.1 IlluminanceTargetLevel Attribute**

6993 The *IlluminanceTargetLevel* attribute specifies the target illuminance level. This target level is taken as the centre of  
 6994 a ‘dead band’, which must be sufficient in width, with hysteresis bands at both top and bottom, to provide reliable  
 6995 notifications without ‘chatter’. Such a dead band and hysteresis bands must be provided by any implementation of this  
 6996 cluster. (N.B. Manufacturer specific attributes may be provided to configure these).

6997 *IlluminanceTargetLevel* represents illuminance in Lux (symbol lx) as follows:

6998  $IlluminanceTargetLevel = 10,000 \times \log_{10} \text{Illuminance}$

6999 Where  $1 \text{ lx} \leq \text{Illuminance} \leq 3.576 \text{ Mlx}$ , corresponding to a *MeasuredValue* in the range 0 to 0xffff.

7000 A value of 0xffff indicates that this attribute is not valid.

7001 **4.3.2.4 Commands Received**

7002 No cluster specific commands are received by the server.

7003 **4.3.2.5 Commands Generated**

7004 No cluster specific commands are generated by the server cluster.

7005 **4.3.2.6 Attribute Reporting**

7006 This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and  
 7007 maximum reporting interval settings described in the ZCL Foundation Specification (see 2.4.7). The following  
 7008 attribute shall be reported:

7009 *LevelStatus*

7010 **4.3.3 Client**

7011 The client cluster has no dependencies, specific attributes nor specific commands generated or received.

## 7012 4.4 Temperature Measurement

### 7013 4.4.1 Overview

7014 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
7015 etc.

7016 The server cluster provides an interface to temperature measurement functionality, including configuration and  
7017 provision of notifications of temperature measurements.

#### 7018 4.4.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2241 2370

#### 7019 4.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TMP	Type 2 (server to client)

#### 7020 4.4.1.3 Cluster Identifiers

Identifier	Name
0x0402	Temperature Measurement

## 7021 4.4.2 Server

### 7022 4.4.2.1 Attributes

7023 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
7024 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant nibble specifies the attribute  
7025 set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed  
7026 in Table 4-12.

7027 **Table 4-12. Temperature Measurement Attribute Sets**

Attribute Set Identifier	Description
0x000	Temperature Measurement Information

#### 7028 4.4.2.1.1 Temperature Measurement Information Attribute Set

7029 The Temperature Measurement Information attribute set contains the attributes summarized in Table 4-13.

7030

**Table 4-13. Temperature Measurement Information Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0000	<i>MeasuredValue</i>	int16	<i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i>	RP	0xffff <sup>47</sup>	M
0x0001	<i>MinMeasuredValue</i>	int16	0x954d – 0x7ffe	R	0x8000	M
0x0002	<i>MaxMeasuredValue</i>	int16	0x954e – 0x7fff	R	0x8000	M
0x0003	<i>Tolerance</i>	uint16	0x0000 – 0x0800	R <sup>48</sup>	-	O

7031 **4.4.2.1.1.1 *MeasuredValue* Attribute**

7032 *MeasuredValue* represents the temperature in degrees Celsius as follows:

7033  $MeasuredValue = 100 \times \text{temperature in degrees Celsius.}$

7034 Where  $-273.15^{\circ}\text{C} \leq \text{temperature} \leq 327.67^{\circ}\text{C}$ , corresponding to a *MeasuredValue* in the range 0x954d to 0x7fff.

7035 The maximum resolution this format allows is 0.01 °C.

7036 A *MeasuredValue* of 0x8000 indicates that the temperature measurement is unknown, otherwise the range SHALL be  
 7037 as described in 4.1.3.

7038 *MeasuredValue* is updated continuously as new measurements are made. *MinMeasuredValue* and *MaxMeasuredValue*  
 7039 define the range of the sensor.

7040 **4.4.2.1.1.2 *MinMeasuredValue* Attribute**

7041 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured.

7042 A *MinMeasuredValue* of 0x8000 indicates that the minimum value is unknown. See 4.1.3 for more details.

7043 **4.4.2.1.1.3 *MaxMeasuredValue* Attribute**

7044 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured.

7045 See 4.1.3 for more details. A *MaxMeasuredValue* of 0x8000 indicates that the maximum value is unknown.

7046 **4.4.2.1.1.4 *Tolerance* Attribute**

7047 See 4.1.3.

7048 **4.4.2.2 Commands**

7049 No cluster specific commands are generated or received by the server cluster.

7050 **4.4.2.3 Attribute Reporting**

7051 This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and

7052 maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7).

7053 The following attributes shall be reported:

7054 *MeasuredValue*<sup>49</sup>

<sup>47</sup> CCB 2370 define what a default value is

<sup>48</sup> CCB 2241 Tolerance is not required to be reportable if supported

<sup>49</sup> CCB 2241 Tolerance is not required to be reportable if supported

7055 **4.4.3 Client**

7056 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7057 **4.5 Pressure Measurement**

7058 **4.5.1 Overview**

7059 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
7060 etc.

7061 The server cluster provides an interface to pressure measurement functionality, including configuration and provision  
7062 of notifications of pressure measurements.

7063 **4.5.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2241 2370

7064 **4.5.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PRS	Type 2 (server to client)

7065 **4.5.1.3 Cluster Identifiers**

Identifier	Name
0x0403	Pressure Measurement

7066 **4.5.2 Server**

7067 **4.5.2.1 Attributes**

7068 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
7069 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
7070 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
7071 are listed in Table 4-14 Pressure Measurement Attribute Sets.

7072 **Table 4-14. Pressure Measurement Attribute Sets**

Attribute Set Identifier	Description
0x000	Pressure Measurement Information

0x001	Extended Pressure Measurement Information
-------	---

7073 **4.5.2.1.1 Pressure Measurement Information Attribute Set**

7074 The Pressure Measurement Information attribute set contains the attributes summarized in Table 4-15.

7075 **Table 4-15. Pressure Measurement Information Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	int16	<i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i>	RP	0x8000 <sup>50</sup>	M
0x0001	<i>MinMeasuredValue</i>	int16	0x8001– 0x7ffe	R	0x8000	M
0x0002	<i>MaxMeasuredValue</i>	int16	0x8002– 0x7fff	R	0x8000	M
0x0003	<i>Tolerance</i>	uint16	0x0000 – 0x0800	R <sup>51</sup>	-	O

7076  
 7077 This set provides for measurements with a fixed maximum resolution of 0.1 kPa.

7078 **4.5.2.1.1.1 MeasuredValue Attribute**

7079 *MeasuredValue* represents the pressure in kPa as follows:

7080  $MeasuredValue = 10 \times \text{Pressure}$

7081 Where  $-3276.7 \text{ kPa} \leq \text{Pressure} \leq 3276.7 \text{ kPa}$ , corresponding to a *MeasuredValue* in the range 0x8001 to 0x7fff.

7082 *MinMeasuredValue* and *MaxMeasuredValue* define the range of the sensor.

7083 A *MeasuredValue* of 0x8000 indicates that the pressure measurement is unknown, otherwise the range SHALL be as  
 7084 described in 4.1.3.

7085 *MeasuredValue* is updated continuously as new measurements are made.

7086 **4.5.2.1.1.2 MinMeasuredValue Attribute**

7087 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of  
 7088 0x8000 means this attribute is not defined. See 4.1.3 for more details.

7089 **4.5.2.1.1.3 MaxMeasuredValue Attribute**

7090 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of  
 7091 0x8000 means this attribute is not defined. See 4.1.3 for more details.

7092 **4.5.2.1.1.4 Tolerance Attribute**

7093 See 4.1.3.

7094 **4.5.2.1.2 Extended Pressure Measurement Information Attribute Set**

7095 The Extended Pressure Measurement Information attribute set contains the attributes summarized in Table 4-16.

<sup>50</sup> CCB 2370 define default values

<sup>51</sup> CCB 2241 Tolerance is not required to be reportable if supported

7096

**Table 4-16. Extended Pressure Measurement Information Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0010	<i>ScaledValue</i>	int16	<i>MinScaledValue</i> – <i>MaxScaledValue</i>	R	0	O Note 1
0x0011	<i>MinScaledValue</i>	int16	0x8001-0x7ffe	R	0x8000	O Note 1
0x0012	<i>MaxScaledValue</i>	int16	0x8002-0x7fff	R	0x8000	O Note 1
0x0013	<i>ScaledTolerance</i>	uint16	0x0000 – 0x0800	R	-	O Note 2
0x0014	<i>Scale</i>	int8	0x81 – 0x7f	R	-	O Note 1

7097 **Note 1:** If any one of these attributes is supported, all four shall be supported.

7098 **Note 2:** If this attribute is supported, all attributes in this set shall be supported.

7099 This attribute set is optional, and allows the range and resolution of measured pressures to be extended beyond those  
7100 catered for by the Pressure Measurement Information Attribute Set, in a way fully backward compatible with devices  
7101 that implement (or can read) only that attribute set.

7102 **4.5.2.1.2.1 ScaledValue Attribute**

7103 *ScaledValue* represents the pressure in Pascals as follows:

7104 
$$ScaledValue = 10^{Scale} \times \text{Pressure in Pa}$$

7105 Where  $-3276.7 \times 10^{Scale} \text{ Pa} \leq \text{Pressure} \leq 3276.7 \times 10^{Scale} \text{ Pa}$  corresponding to a *ScaledValue* in the range 0x8001 to  
7106 0x7fff.

7107 A *ScaledValue* of 0x8000 indicates that the pressure measurement is invalid.

7108 *ScaledValue* is updated continuously as new measurements are made.

7109 **4.5.2.1.2.2 MinScaledValue Attribute**

7110 The *MinScaledValue* attribute indicates the minimum value of *ScaledValue* that can be measured. A value of 0x8000  
7111 means this attribute is not defined

7112 **4.5.2.1.2.3 MaxScaledValue Attribute**

7113 The *MaxScaledValue* attribute indicates the maximum value of *ScaledValue* that can be measured. A value of 0x8000  
7114 means this attribute is not defined.

7115 *MaxScaledValue* shall be greater than *MinScaledValue*.

7116 *MinScaledValue* and *MaxScaledValue* define the range of the sensor.

7117 **4.5.2.1.2.4 ScaledTolerance Attribute**

7118 The *ScaledTolerance* attribute indicates the magnitude of the possible error that is associated with *ScaledValue*. The  
7119 true value is located in the range

7120  $(ScaledValue - ScaledTolerance)$  to  $(ScaledValue + ScaledTolerance)$ .

7121 **4.5.2.1.2.5 Scale Attribute**

7122 The *Scale* attribute indicates the base 10 exponent used to obtain *ScaledValue* (see 4.5.2.1.2.1).



7123 **4.5.2.2 Commands**

7124 No cluster specific commands are generated or received by the server cluster.

7125 **4.5.2.3 Attribute Reporting**

7126 This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and  
 7127 maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7).  
 7128 The following attributes shall be reportable:

7129 *MeasuredValue*<sup>52</sup>

7130 If the Extended Pressure Measurement Information attribute set is implemented, it is recommended that the following  
 7131 attributes are also reportable:

7132 *ScaledValue*

7133 *ScaledTolerance*

7134 **4.5.3 Client**

7135 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7136 **4.6 Flow Measurement**

---

7137 **4.6.1 Overview**

7138 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7139 etc.

7140 The server cluster provides an interface to flow measurement functionality, including configuration and provision of  
 7141 notifications of flow measurements.

7142 **4.6.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2241 2370

7143 **4.6.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	FLW	Type 2 (server to client)

7144 **4.6.1.3 Cluster Identifiers**

Identifier	Name
0x0404	Flow Measurement

<sup>52</sup> CCB 2241 Tolerance is not required to be reportable if supported

7145 **4.6.2 Server**7146 **4.6.2.1 Attributes**

7147 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
7148 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
7149 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
7150 for are listed in Table 4-17.

7151 **Table 4-17. Flow Measurement Attribute Sets**

Attribute Set Identifier	Description
0x000	Flow Measurement Information

7152 **4.6.2.1.1 Flow Measurement Information Attribute Set**

7153 The Flow Measurement Information attribute set contains the attributes summarized in Table 4-18.

7154 **Table 4-18. Flow Measurement Information Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	uint16	<i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i>	RP	0xffff <sup>53</sup>	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0000 – 0xfffd	R	0xffff	M
0x0002	<i>MaxMeasuredValue</i>	uint16	0x0001 – 0xfffe	R	0xffff	M
0x0003	<i>Tolerance</i>	uint16	0x0000 – 0x0800	R <sup>54</sup>		O

7155 **4.6.2.1.1.1 MeasuredValue Attribute**

7156 *MeasuredValue* represents the flow in m<sup>3</sup>/h as follows:

7157  $MeasuredValue = 10 \times Flow$

7158 Where  $0 \text{ m}^3/\text{h} \leq Flow \leq 6,553.4 \text{ m}^3/\text{h}$ , corresponding to a *MeasuredValue* in the range 0 to 0xffffe.

7159 The maximum resolution this format allows is 0.1 m<sup>3</sup>/h.

7160 *MinMeasuredValue* and *MaxMeasuredValue* define the range of the sensor.

7161 A *MeasuredValue* of 0xffff indicates that the pressure measurement is unknown, otherwise the range SHALL be as  
7162 described in 4.1.3.

7163 *MeasuredValue* is updated continuously as new measurements are made.

7164 **4.6.2.1.1.2 MinMeasuredValue Attribute**

7165 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of  
7166 0xffff means this attribute is not defined. See 4.1.3 for more details.

<sup>53</sup> CCB 2370 better define default values

<sup>54</sup> CCB 2241 Tolerance is not required to be reportable if supported

7167 **4.6.2.1.1.3 MaxMeasuredValue Attribute**

7168 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of  
 7169 0xffff means this attribute is not defined. See 4.1.3 for more details.

7170 **4.6.2.1.1.4 Tolerance Attribute**

7171 See 4.1.3.

7172 **4.6.2.2 Commands**

7173 No cluster specific commands are generated or received by the server cluster.

7174 **4.6.2.3 Attribute Reporting**

7175 This cluster shall support attribute reporting using the Report Attributes command and according to the minimum and  
 7176 maximum reporting interval and reportable change settings described in the ZCL Foundation specification (see 2.4.7).  
 7177 The following attributes shall be reported:

7178 *MeasuredValue*<sup>55</sup>

7179 **4.6.3 Client**

7180 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7181 **4.7 Water Content<sup>56</sup> Measurement**

7182 **4.7.1 Overview**

7183 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7184 etc.

7185 This is a base cluster. The server cluster provides an interface to water content measurement functionality. The  
 7186 measurement is reportable and may be configured for reporting. Water content measurements include, but are not  
 7187 limited to, leaf wetness, relative humidity, and soil moisture.

7188 **4.7.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2241

7189 **4.7.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	RH	Type 2 (server to client)

<sup>55</sup> CCB 2241 Tolerance is not required to be reportable if supported

<sup>56</sup> Formerly was Relative Humidity, now a generic base cluster with multiple Cluster Ids

7190 **4.7.1.3 Cluster Identifiers**

Identifier	Name	Description
0x0405	Relative Humidity	Percentage of water in the air
0x0407	Leaf Wetness	Percentage of water on the leaves of plants
0x0408	Soil Moisture	Percentage of water in the soil

7191 **4.7.2 Server**7192 **4.7.2.1 Attributes**7193 **Table 4-19. Attributes of the Water Content cluster**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	uint16	<i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i>	RP	0xffff	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0000 – 0x270f	R	0xffff	M
0x0002	<i>MaxMeasuredValue</i>	uint16	0x0001 – 0x2710	R	0xffff	M
0x0003	<i>Tolerance</i>	uint16	0x0000 – 0x0800	R <sup>57</sup>		O

7194 **4.7.2.1.1 *MeasuredValue* Attribute**7195 *MeasuredValue* represents the water content in % as follows:7196  $MeasuredValue = 100 \times \text{water content}$ 7197 Where  $0\% \leq \text{water content} \leq 100\%$ , corresponding to a *MeasuredValue* in the range 0 to 0x2710.

7198 The maximum resolution this format allows is 0.01%.

7199 *MinMeasuredValue* and *MaxMeasuredValue* define the range of the sensor.7200 A *MeasuredValue* of 0xffff indicates that the measurement is unknown, otherwise the range SHALL be as described  
7201 in 4.1.3.7202 *MeasuredValue* is updated continuously as new measurements are made.7203 **4.7.2.1.2 *MinMeasuredValue* Attribute**7204 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that can be measured. A value of  
7205 0xffff means this attribute is not defined. See 4.1.3 for more details.7206 **4.7.2.1.3 *MaxMeasuredValue* Attribute**7207 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that can be measured. A value of  
7208 0xffff means this attribute is not defined. See 4.1.3 for more details.<sup>57</sup> CCB 2241 Tolerance is not required to be reportable if supported

7209 **4.7.2.1.4 Tolerance Attribute**

7210 See 4.1.3.

7211 **4.7.2.2 Commands**

7212 No cluster specific commands are received or generated by the server cluster.

7213 **4.7.3 Client**

7214 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7215 **4.8 Occupancy Sensing**

---

7216 **4.8.1 Overview**

7217 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7218 etc.

7219 The server cluster provides an interface to occupancy sensing functionality, including configuration and provision of  
 7220 notifications of occupancy status.

7221 **4.8.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	Physical Contact Occupancy feature with mandatory <i>OccupancySensorTypeBitmap</i>

7222 **4.8.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	OCC	Type 2 (server to client)

7223 **4.8.1.3 Cluster Identifiers**

Identifier	Name
0x0406	Occupancy Sensing

7224 **4.8.2 Server**

7225 **4.8.2.1 Attributes**

7226 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 7227 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 7228 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 7229 are listed in Table 4-20.

7230

**Table 4-20. Occupancy Sensor Attribute Sets**

Attribute Set Identifier	Description
0x000	Occupancy sensor information
0x001	PIR configuration
0x002	Ultrasonic configuration
0x003	Physical contact configuration <sup>58</sup>

7231 **4.8.2.1.1 Occupancy Sensor Information Set**

7232 The occupancy sensor information attribute set contains the attributes summarized in Table 4-21.

7233 **Table 4-21. Occupancy Sensor Information Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0000	<i>Occupancy</i>	map8	0b0000 000x	RP	-	M
0x0001	<i>OccupancySensorType</i>	enum8		R	MS	M
0x0002	<i>OccupancySensorTypeBitmap</i> <sup>59</sup>	map8	0000 0xxx	R	-	M

7234 **4.8.2.1.1.1 Occupancy Attribute**

7235 The *Occupancy* attribute is a bitmap.

7236 Bit 0 specifies the sensed occupancy as follows: 1 = occupied, 0 = unoccupied.

7237 All other bits are reserved.

7238 **4.8.2.1.1.2 OccupancySensorType Attribute**

7239 The *OccupancySensorType* attribute specifies the type of the occupancy sensor. This attribute shall be set to one of the non-reserved values listed in Table 4-22.

7241 **Table 4-22. Values of the *OccupancySensorType* Attribute**

Attribute Value	Description
0x00	PIR
0x01	Ultrasonic
0x02	PIR and ultrasonic
0x03	Physical contact <sup>60</sup>

7242 **4.8.2.1.1.3 OccupancySensorTypeBitmap Attribute**

<sup>58</sup> NFR Physical Contact Occupancy Sensor

<sup>59</sup> NFR Physical Contact Occupancy Sensor

<sup>60</sup> NFR Physical Contact Occupancy Sensor

7243 The *OccupancySensorTypeBitmap* attribute specifies the types of the occupancy sensor, as listed below; a ‘1’ in each  
 7244 bit position indicates this type is implemented.

7245 **Table 4-23. The *OccupancySensorTypeBitmap* Attribute**

Bit	Description
Bit0	PIR
Bit1	Ultrasonic
Bit2	Physical contact

7246 The value of the *OccupancySensorTypeBitmap* attribute and the *OccupancySensorType* attribute SHALL be aligned  
 7247 as defined below.

7248 **Table 4-24. Mapping between *OccupancySensorType* and *OccupancySensorTypeBitmap* Attributes**

Description	<i>OccupancySensorType</i> attribute	<i>OccupancySensorTypeBitmap</i> attribute
PIR	0x00	0000 0001
Ultrasonic	0x01	0000 0010
PIR and ultrasonic	0x02	0000 0011
Physical contact and PIR	0x00	0000 0101
Physical contact and ultrasonic	0x01	0000 0110
Physical contact and PIR and ultrasonic	0x02	0000 0111

7249

7250 **4.8.2.1.2 PIR Configuration Set**

7251 The PIR sensor configuration attribute set contains the attributes summarized in Table 4-25.

7252 **Table 4-25. Attributes of the PIR Configuration Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0010	<i>PIROccupiedToUnoccupiedDelay</i>	uint16	0x00 – 0xfffe	RW	0x00	O
0x0011	<i>PIRUnoccupiedToOccupiedDelay</i>	uint16	0x00 – 0xfffe	RW	0x00	O
0x0012	<i>PIRUnoccupiedToOccupiedThreshold</i>	uint8	0x01 – 0xfe	RW	0x01	O

7253 **4.8.2.1.2.1 *PIROccupiedToUnoccupiedDelay* Attribute**

7254 The *PIROccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the  
7255 PIR sensor changes to its unoccupied state after the last detection of movement in the sensed area.

7256 **4.8.2.1.2.2 *PIRUnoccupiedToOccupiedDelay* Attribute**

7257 The *PIRUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds, before the  
7258 PIR sensor changes to its occupied state after the detection of movement in the sensed area. This attribute is mandatory  
7259 if the *PIRUnoccupiedToOccupiedThreshold* attribute is implemented.

7260 **4.8.2.1.2.3 *PIRUnoccupiedToOccupiedThreshold* Attribute**

7261 The *PIRUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of movement detection  
7262 events that must occur in the period *PIRUnoccupiedToOccupiedDelay*, before the PIR sensor changes to its occupied  
7263 state. This attribute is mandatory if the *PIRUnoccupiedToOccupiedDelay* attribute is implemented.

7264 **4.8.2.1.3 Ultrasonic Configuration Set**

7265 The ultrasonic sensor configuration attribute set contains the attributes summarized in Table 4-26.

7266 **Table 4-26. Attributes of the Ultrasonic Configuration Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
0x0020	<i>UltrasonicOccupiedToUnoccupiedDelay</i>	uint16	0x0000 – 0xfffe	RW	0x00	O
0x0021	<i>UltrasonicUnoccupiedToOccupiedDelay</i>	uint16	0x0000 – 0xfffe	RW	0x00	O
0x0022	<i>UltrasonicUnoccupiedToOccupiedThreshold</i>	uint8	0x01 – 0xfe	RW	0x01	O

7267 **4.8.2.1.3.1 *UltrasonicOccupiedToUnoccupiedDelay* Attribute**

7268 The *UltrasonicOccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds,  
7269 before the Ultrasonic sensor changes to its unoccupied state after the last detection of movement in the sensed area.

7270 **4.8.2.1.3.2 *UltrasonicUnoccupiedToOccupiedDelay* Attribute**

7271 The *UltrasonicUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in seconds,  
7272 before the Ultrasonic sensor changes to its occupied state after the detection of movement in the sensed area. This  
7273 attribute is mandatory if the *UltrasonicUnoccupiedToOccupiedThreshold* attribute is implemented.

7274 **4.8.2.1.3.3 *UltrasonicUnoccupiedToOccupiedThreshold* Attribute**

7275 The *UltrasonicUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of movement  
7276 detection events that must occur in the period *UltrasonicUnoccupiedToOccupiedDelay*, before the Ultrasonic sensor  
7277 changes to its occupied state. This attribute is mandatory if the *UltrasonicUnoccupiedToOccupiedDelay* attribute is  
7278 implemented.

7279 **4.8.2.1.4 Physical Contact Configuration Set<sup>61</sup>**

7280 The physical contact configuration attribute set contains the attributes summarized below.

7281 **Table 4-27. Attributes of the Physical Contact Configuration Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
----	------	------	-------	-----	-----	----

<sup>61</sup> NFR Physical Contact Occupancy Sensor



0x0030	<i>PhysicalContactOccupiedToUnoccupiedDelay</i>	uint16	0x0000 to 0xffff	RW	0x0000	O
0x0031	<i>PhysicalContactUnoccupiedToOccupiedDelay</i>	uint16	0x0000 to 0xffff	RW	0x0000	O
0x0032	<i>PhysicalContactUnoccupiedToOccupiedThreshold</i>	uint8	0x01 to 0xfe	RW	0x01	O

7282 **4.8.2.1.4.1 PhysicalContactOccupiedToUnoccupiedDelay Attribute**

7283 The *PhysicalContactOccupiedToUnoccupiedDelay* attribute is 16 bits in length and specifies the time delay, in  
 7284 seconds, before the physical contact occupancy sensor changes to its unoccupied state after detecting the unoccupied  
 7285 event. The value of 0xffff indicates the sensor does not report occupied to unoccupied transition.

7286 **4.8.2.1.4.2 PhysicalContactUnoccupiedToOccupiedDelay Attribute**

7287 The *PhysicalContactUnoccupiedToOccupiedDelay* attribute is 16 bits in length and specifies the time delay, in  
 7288 seconds, before the physical contact sensor changes to its occupied state after the detection of the occupied event.  
 7289 The value of 0xffff indicates the sensor does not report unoccupied to occupied transition.

7290 **4.8.2.1.4.3 PhysicalContactUnoccupiedToOccupiedThreshold Attribute**

7291 The *PhysicalContactUnoccupiedToOccupiedThreshold* attribute is 8 bits in length and specifies the number of  
 7292 movement detection events that must occur in the period *PhysicalContactUnoccupiedToOccupiedDelay*, before the  
 7293 PIR sensor changes to its occupied state. This attribute is mandatory if the  
 7294 *PhysicalContactUnoccupiedToOccupiedDelay* attribute is implemented.

7295 **4.8.2.2 Commands**

7296 No cluster specific commands are received by the server cluster. No cluster specific commands are generated by the  
 7297 server cluster.

7298 **4.8.3 Client**

7299 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7300 **4.9 Electrical Measurement**

---

7301 **4.9.1 Overview**

7302 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7303 etc.

7304 This cluster provides a mechanism for querying data about the electrical properties as measured by the device. This  
 7305 cluster may be implemented on any device type and be implemented on a per-endpoint basis. For example, a power  
 7306 strip device could represent each outlet on a different endpoint and report electrical information for each individual  
 7307 outlet. The only caveat is that if you implement an attribute that has an associated multiplier and divisor, then you  
 7308 must implement the associated multiplier and divisor attributes. For example if you implement DCVoltage, you must  
 7309 also implement DCVoltageMultiplier and DCVoltageDivisor.

7310 If you are interested in reading information about the power supply or battery level on the device, please see the Power  
 7311 Configuration cluster.

7312 **4.9.1.1 Revision History**

Rev	Description
-----	-------------

1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2236

7313 **4.9.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	EMR	Type 1 (client to server)

7314 **4.9.1.3 Cluster Identifiers**

Identifier	Name
0x0b04	Electrical Measurement

7315 **4.9.1.4 Formatting**

7316 Most measurement values have an associated multiplier and divisor attribute. Multiplier attributes provide a value to  
7317 be multiplied against a raw or uncompensated measurement value. Divisor attributes provide a value to divide the  
7318 results of applying a multiplier attribute against a raw or uncompensated measurement value. If a multiplier or divisor  
7319 attribute is present, its corresponding divisor or multiplier attribute shall be implemented as well.

7320 **4.9.2 Server**

7321 **4.9.2.1 Dependencies**

7322 For the alarm functionality in this cluster to be operational, any endpoint that implements the Electrical Measurement  
7323 server cluster shall also implement the Alarms server cluster.

7324 **4.9.2.2 Attributes**

7325 The server side of this cluster contains certain attributes associated with the electrical properties and configuration, as  
7326 shown in Table 4-28.

7327 **Table 4-28. Attributes of the Electrical Measurement Cluster**

Attribute Set Identifier	Description
0x00	Basic Information
0x01	DC Measurement
0x02	DC Formatting
0x03	AC (Non-phase Specific) Measurements
0x04	AC (Non-phase Specific) Formatting
0x05	AC (Single Phase or Phase A) Measurements
0x06	AC Formatting

Attribute Set Identifier	Description
0x07	DC Manufacturer Threshold Alarms
0x08	AC Manufacturer Threshold Alarms
0x09	AC Phase B Measurements
0x0a	AC Phase C Measurements

7328 **4.9.2.2.1 Basic Information**

7329 **Table 4-29. Electrical Measurement Cluster Basic Information**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasurementType</i>	map32	0x00000000 – 0xffffFFFF	R	0x00000000	M

7330 **4.9.2.2.1.1 MeasurementType**

7331 This attribute indicates a device’s measurement capabilities. This will be indicated by setting the desire measurement  
 7332 bits to 1, as mentioned in Table 4-30 and DC Measurement

7333 Table 4-31. This attribute will be used client devices to determine what all attribute is supported by the meter. Unused  
 7334 bits should be set to zero.

7335 **Table 4-30. MeasurementType Attribute**

Bit	Flag Name / Description
0	Active measurement (AC)
1	Reactive measurement (AC)
2	Apparent measurement (AC)
3	Phase A measurement
4	Phase B measurement
5	Phase C measurement
6	DC measurement
7	Harmonics measurement
8	Power quality measurement

7336 **4.9.2.2.2 DC Measurement**7337 **Table 4-31. DC Measurement Attributes**

Id	Name	Type	Range	Acc	Default	M/O
0x0100	<i>DCVoltage</i>	int16	-32767 – 32767	RP	0x8000	O
0x0101	<i>DCVoltageMin</i>	int16	-32767 – 32767	R	0x8000	O
0x0102	<i>DCVoltageMax</i>	int16	-32767 – 32767	R	0x8000	O
0x0103	<i>DCCurrent</i>	int16	-32767 – 32767	RP	0x8000	O
0x0104	<i>DCCurrentMin</i>	int16	-32767 – 32767	R	0x8000	O
0x0105	<i>DCCurrentMax</i>	int16	-32767 – 32767	R	0x8000	O
0x0106	<i>DCPower</i>	int16	-32767 – 32767	RP	0x8000	O
0x0107	<i>DCPowerMin</i>	int16	-32767 – 32767	R	0x8000	O
0x0108	<i>DCPowerMax</i>	int16	-32767 – 32767	R	0x8000	O
0x0109 – 0x01FF	Reserved					

7338 **4.9.2.2.2.1 DCVoltage**

7339 The DCVoltage attribute represents the most recent DC voltage reading in Volts (V). If the voltage cannot be  
7340 measured, a value of 0x8000 is returned.

7341 **4.9.2.2.2.2 DCVoltageMin**

7342 The DCVoltageMin attribute represents the lowest DC voltage value measured in Volts (V). After resetting, this  
7343 attribute will return a value of 0x8000 until a measurement is made.

7344 **4.9.2.2.2.3 DCVoltageMax**

7345 The DCVoltageMax attribute represents the highest DC voltage value measured in Volts (V). After resetting, this  
7346 attribute will return a value of 0x8000 until a measurement is made.

7347 **4.9.2.2.2.4 DCCurrent**

7348 The DCCurrent attribute represents the most recent DC current reading in Amps (A). If the current cannot be  
7349 measured, a value of 0x8000 is returned.

7350 **4.9.2.2.2.5 DCCurrentMin**

7351 The DCCurrentMin attribute represents the lowest DC current value measured in Amps (A). After resetting, this  
7352 attribute will return a value of 0x8000 until a measurement is made.

7353 **4.9.2.2.2.6 DCCurrentMax**

7354 The DCCurrentMax attribute represents the highest DC current value measured in Amps (A). After resetting, this  
7355 attribute will return a value of 0x8000 until a measurement is made.

7356 **4.9.2.2.2.7** ***DCPower***

7357 The DCPower attribute represents the most recent DC power reading in Watts (W). If the power cannot be measured,  
 7358 a value of 0x8000 is returned.

7359 **4.9.2.2.2.8** ***DCPowerMin***

7360 The DCPowerMin attribute represents the lowest DC power value measured in Watts (W). After resetting, this  
 7361 attribute will return a value of 0x8000 until a measurement is made.

7362 **4.9.2.2.2.9** ***DCPowerMax***

7363 The DCPowerMax attribute represents the highest DC power value measured in Watts (W). After resetting, this  
 7364 attribute will return a value of 0x8000 until a measurement is made.

7365 **4.9.2.2.3** ***DC Formatting***

7366 **Table 4-32. DC Formatting Attributes**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Default</b>	<b>M/O</b>
0x0200	<i>DCVoltageMultiplier</i>	uint16	0x0001 – 0xffff	RP <sup>62</sup>	0x0001	O
0x0201	<i>DCVoltageDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0202	<i>DCCurrentMultiplier</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0203	<i>DCCurrentDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0204	<i>DCPowerMultiplier</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0205	<i>DCPowerDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O

7367 **4.9.2.2.3.1** ***DCVoltageMultiplier***

7368 The *DCVoltageMultiplier* provides a value to be multiplied against the *DCVoltage*, *DCVoltageMin*, and  
 7369 *DCVoltageMax* attributes. This attribute must be used in conjunction with the *DCVoltageDivisor* attribute. 0x0000 is  
 7370 an invalid value for this attribute.

7371 **4.9.2.2.3.2** ***DCVoltageDivisor***

7372 The *DCVoltageDivisor* provides a value to be divided against the *DCVoltage*, *DCVoltageMin*, and *DCVoltageMax*  
 7373 attributes. This attribute must be used in conjunction with the *DCVoltageMultiplier* attribute. 0x0000 is an invalid  
 7374 value for this attribute.

7375 **4.9.2.2.3.3** ***DCCurrentMultiplier***

7376 The *DCCurrentMultiplier* provides a value to be multiplied against the *DCCurrent*, *DCCurrentMin*, and  
 7377 *DCCurrentMax* attributes. This attribute must be used in conjunction with the *DCCurrentDivisor* attribute. 0x0000 is  
 7378 an invalid value for this attribute.

7379 **4.9.2.2.3.4** ***DCCurrentDivisor***

<sup>62</sup> CCB 2236 for all multipliers and divisors that may change, though less frequently than the values

7380 The *DCCurrentDivisor* provides a value to be divided against the *DCCurrent*, *DCCurrentMin*, and *DCCurrentMax*  
7381 attributes. This attribute must be used in conjunction with the *DCCurrentMultiplier* attribute. 0x0000 is an invalid  
7382 value for this attribute.

7383 **4.9.2.2.3.5 DCPowerMultiplier**

7384 The *DCPowerMultiplier* provides a value to be multiplied against the *DCPower*, *DCPowerMin*, and *DCPowerMax*  
7385 attributes. This attribute must be used in conjunction with the *DCPowerDivisor* attribute. 0x0000 is an invalid value  
7386 for this attribute.

7387 **4.9.2.2.3.6 DCPowerDivisor**

7388 The *DCPowerDivisor* provides a value to be divided against the *DCPower*, *DCPowerMin*, and *DCPowerMax*  
7389 attributes. This attribute must be used in conjunction with the *DCPowerMultiplier* attribute. 0x0000 is an invalid value  
7390 for this attribute.

7391 **4.9.2.2.4 AC (Non-phase Specific) Measurements**

7392 **Table 4-33. AC (Non-phase Specific) Measurement Attributes**

Id	Name	Type	Range	Acc	Default	M/O
0x0300	<i>ACFrequency</i>	uint16	0x0000 – 0xffff	RP <sup>63</sup>	0xffff	O
0x0301	<i>ACFrequencyMin</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0302	<i>ACFrequencyMax</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0303	<i>NeutralCurrent</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0304	<i>TotalActivePower</i>	int32	-8,388,607–8,388,607	RP	-	O
0x0305	<i>TotalReactivePower</i>	int32	-8,388,607–8,388,607	RP	-	O
0x0306	<i>TotalApparentPower</i>	uint32	0x000000–0xffffFF	RP	-	O
0x0307	<i>Measured1stHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0308	<i>Measured3rdHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0309	<i>Measured5thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x030a	<i>Measured7thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x030b	<i>Measured9thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x030c	<i>Measured11thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x030d	<i>MeasuredPhase1stHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x030e	<i>MeasuredPhase3rdHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O

<sup>63</sup> CCB 2236 for all changing values

Id	Name	Type	Range	Acc	Default	M/O
0x030f	<i>MeasuredPhase5thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0310	<i>MeasuredPhase7thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0311	<i>MeasuredPhase9thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0312	<i>MeasuredPhase11thHarmonicCurrent</i>	int16	-32768 – 32767	RP	0x8000	O

7393 **4.9.2.2.4.1 ACFrequency**

7394 The *ACFrequency* attribute represents the most recent AC Frequency reading in Hertz (Hz). If the frequency cannot  
 7395 be measured, a value of 0xffff is returned.

7396 **4.9.2.2.4.2 ACFrequencyMin**

7397 The *ACFrequencyMin* attribute represents the lowest AC Frequency value measured in Hertz (Hz). After resetting,  
 7398 this attribute will return a value of 0xffff until a measurement is made.

7399 **4.9.2.2.4.3 ACFrequencyMax**

7400 The *ACFrequencyMax* attribute represents the highest AC Frequency value measured in Hertz (Hz). After resetting,  
 7401 this attribute will return a value of 0xffff until a measurement is made.

7402 **4.9.2.2.4.4 NeutralCurrent**

7403 The *NeutralCurrent* attribute represents the AC neutral (Line-Out) current value at the moment in time the attribute  
 7404 is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0xffff is returned.

7405 **4.9.2.2.4.5 TotalActivePower**

7406 Active power represents the current demand of active power delivered or received at the premises, in kW. Positive  
 7407 values indicate power delivered to the premises where negative values indicate power received from the premises. In  
 7408 case if device is capable of measuring multi elements or phases then this will be net active power value.

7409 **4.9.2.2.4.6 TotalReactivePower**

7410 Reactive power represents the current demand of reactive power delivered or received at the premises, in kVAr.  
 7411 Positive values indicate power delivered to the premises where negative values indicate power received from the  
 7412 premises. In case if device is capable of measuring multi elements or phases then this will be net reactive power value.

7413 **4.9.2.2.4.7 TotalApparentPower**

7414 Represents the current demand of apparent power, in kVA. In case if device is capable of measuring multi elements  
 7415 or phases then this will be net apparent power value.

7416 **4.9.2.2.4.8 MeasuredNthHarmonicCurrent Attributes**

7417 The *Measured1stHarmonicCurrent* through *MeasuredNthHarmonicCurrent* attributes represent the most recent N<sup>th</sup>  
 7418 harmonic current reading in an AC frequency. The unit for this measurement is  $10^{NthHarmonicCurrentMultiplier}$   
 7419 amperes. If *NthHarmonicCurrentMultiplier* is not implemented the unit is in amperes. If the N<sup>th</sup> harmonic current  
 7420 cannot be measured a value of 0x8000 is returned. A positive value indicates the measured N<sup>th</sup> harmonic current is  
 7421 positive, and a negative value indicates that the measured N<sup>th</sup> harmonic current is negative.

7422 **4.9.2.2.4.9 MeasuredPhaseNthHarmonicCurrent Attributes**

7423 The *MeasuredPhase1stHarmonicCurrent* through *MeasuredPhaseNthHarmonicCurrent* attributes represent the most  
 7424 recent phase of the N<sup>th</sup> harmonic current reading in an AC frequency. The unit for this measurement is  $10^{\wedge}$   
 7425 *PhaseNthHarmonicCurrentMultiplier* degree. If *PhaseNthHarmonicCurrentMultiplier* is not implemented the unit is  
 7426 in degree. If the phase of the N<sup>th</sup> harmonic current cannot be measured a value of 0x8000 is returned. A positive value  
 7427 indicates the measured phase of the N<sup>th</sup> harmonic current is prehurry, and a negative value indicates that the measured  
 7428 phase of the N<sup>th</sup> harmonic current is lagging.

#### 7429 4.9.2.2.5 AC (Non-phase Specific) Formatting

7430 Table 4-34. AC (Non-phase Specific) Formatting Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0400	<i>ACFrequencyMultiplier</i>	uint16	0x0001 – 0xffff	RP <sup>64</sup>	0x0001	O
0x0401	<i>ACFrequencyDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0402	<i>PowerMultiplier</i>	uint32	0x000000 – 0xffffFF	RP	0x000001	O
0x0403	<i>PowerDivisor</i>	uint32	0x00000 – 0xffffFF	RP	0x000001	O
0x0404	<i>HarmonicCurrentMultiplier</i>	int8	-127 – 127	RP	0x00	O
0x0405	<i>PhaseHarmonicCurrentMultiplier</i>	int8	-127 – 127	RP	0x00	O

##### 7431 4.9.2.2.5.1 *ACFrequencyMultiplier*

7432 Provides a value to be multiplied against the *ACFrequency* attribute. This attribute must be used in conjunction with  
 7433 the *ACFrequencyDivisor* attribute. 0x0000 is an invalid value for this attribute.

##### 7434 4.9.2.2.5.2 *ACFrequencyDivisor*

7435 Provides a value to be divided against the *ACFrequency* attribute. This attribute must be used in conjunction with the  
 7436 *ACFrequencyMultiplier* attribute. 0x0000 is an invalid value for this attribute.

##### 7437 4.9.2.2.5.3 *PowerMultiplier*

7438 Provides a value to be multiplied against a raw or uncompensated sensor count of power being measured by the  
 7439 metering device. If present, this attribute must be applied against all power/demand values to derive the delivered and  
 7440 received values expressed in the specified units. This attribute must be used in conjunction with the *PowerDivisor*  
 7441 attribute.

##### 7442 4.9.2.2.5.4 *PowerDivisor*

7443 Provides a value to divide against the results of applying the *Multiplier* attribute against a raw or uncompensated  
 7444 sensor count of power being measured by the metering device. If present, this attribute must be applied against all  
 7445 demand/power values to derive the delivered and received values expressed in the specified units. This attribute must  
 7446 be used in conjunction with the *PowerMultiplier* attribute.

##### 7447 4.9.2.2.5.5 *HarmonicCurrentMultiplier*

7448 Represents the unit value for the *MeasuredNthHarmonicCurrent* attribute in the format  
 7449 *MeasuredNthHarmonicCurrent* \*  $10^{\wedge}$  *HarmonicCurrentMultiplier* amperes.

<sup>64</sup> CCB 2236 for all multipliers and divisors that may change, though less frequently than the values



7450 **4.9.2.2.5.6 PhaseHarmonicCurrentMultiplier**

7451 Represents the unit value for the *MeasuredPhaseNthHarmonicCurrent* attribute in the format  
 7452 *MeasuredPhaseNthHarmonicCurrent* \* 10 ^ *PhaseHarmonicCurrentMultiplier* degrees.

7453 **4.9.2.2.6 AC (Single Phase or Phase A) Measurements**

7454 Table 4-35. AC (Single Phase or Phase A) Measurement Attributes

Id	Name	Type	Range	Acc	Default	M/O
0x0501	<i>LineCurrent</i>	uint16	0x0000 – 0xffff	RP <sup>65</sup>	0xffff	O
0x0502	<i>ActiveCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0503	<i>ReactiveCurrent</i>	int16	-32768 – 32767	RP	0x8000	O
0x0505	<i>RMSVoltage</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0506	<i>RMSVoltageMin</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0507	<i>RMSVoltageMax</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0508	<i>RMSCurrent</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0509	<i>RMSCurrentMin</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x050a	<i>RMSCurrentMax</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x050b	<i>ActivePower</i>	int16	-32768 – 32767	RP	0x8000	O
0x050c	<i>ActivePowerMin</i>	int16	-32768 – 32767	R	0x8000	O
0x050d	<i>ActivePowerMax</i>	int16	-32768 – 32767	R	0x8000	O
0x050e	<i>ReactivePower</i>	int16	-32768 – 32767	RP	0x8000	O
0x050f	<i>ApparentPower</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0510	<i>PowerFactor</i>	int8	-100 to +100	R	0x00	O
0x0511	<i>AverageRMSVoltageMeasurementPeriod</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0512	<i>AverageRMSOverVoltageCounter</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0513	<i>AverageRMSUnderVoltageCounter</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0514	<i>RMSExtremeOverVoltagePeriod</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0515	<i>RMSExtremeUnderVoltagePeriod</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0516	<i>RMSVoltageSagPeriod</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0517	<i>RMSVoltageSwellPeriod</i>	uint16	0x0000 – 0xffff	RW	0x0000	O

7455 **4.9.2.2.6.1 LineCurrent**

<sup>65</sup> CCB 2236 for all changing values

7456 Represents the single phase or Phase A, AC line current (Square root of active and reactive current) value at the  
7457 moment in time the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000  
7458 is returned.

7459 **4.9.2.2.6.2      *ActiveCurrent***

7460 Represents the single phase or Phase A, AC active/resistive current value at the moment in time the attribute is read,  
7461 in Amps (A). Positive values indicate power delivered to the premises where negative values indicate power received  
7462 from the premises.

7463 **4.9.2.2.6.3      *ReactiveCurrent***

7464 Represents the single phase or Phase A, AC reactive current value at the moment in time the attribute is read, in Amps  
7465 (A). Positive values indicate power delivered to the premises where negative values indicate power received from the  
7466 premises.

7467 **4.9.2.2.6.4      *RMSVoltage***

7468 Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of  
7469 0xffff is returned.

7470 **4.9.2.2.6.5      *RMSVoltageMin***

7471 Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
7472 0xffff until a measurement is made.

7473 **4.9.2.2.6.6      *RMSVoltageMax***

7474 Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
7475 0xffff until a measurement is made.

7476 **4.9.2.2.6.7      *RMSCurrent***

7477 Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is  
7478 returned.

7479 **4.9.2.2.6.8      *RMSCurrentMin***

7480 Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
7481 0xffff until a measurement is made.

7482 **4.9.2.2.6.9      *RMSCurrentMax***

7483 Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
7484 0xffff until a measurement is made.

7485 **4.9.2.2.6.10     *ActivePower***

7486 Represents the single phase or Phase A, current demand of active power delivered or received at the premises, in Watts  
7487 (W). Positive values indicate power delivered to the premises where negative values indicate power received from the  
7488 premises.

7489 **4.9.2.2.6.11     *ActivePowerMin***

7490 Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
7491 0x8000 until a measurement is made.

7492 **4.9.2.2.6.12     *ActivePowerMax***

7493 Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
7494 0x8000 until a measurement is made.

- 7495 **4.9.2.2.6.13**      ***ReactivePower***
- 7496 Represents the single phase or Phase A, current demand of reactive power delivered or received at the premises, in  
 7497 VAR. Positive values indicate power delivered to the premises where negative values indicate power received from  
 7498 the premises.
- 7499 **4.9.2.2.6.14**      ***ApparentPower***
- 7500 Represents the single phase or Phase A, current demand of apparent (Square root of active and reactive power) power,  
 7501 in VA.
- 7502 **4.9.2.2.6.15**      ***PowerFactor***
- 7503 Contains the single phase or PhaseA, Power Factor ratio in 1/100ths.
- 7504 **4.9.2.2.6.16**      ***AverageRMSVoltageMeasurementPeriod***
- 7505 The Period in seconds that the RMS voltage is averaged over.
- 7506 **4.9.2.2.6.17**      ***AverageRMSOverVoltageCounter***
- 7507 The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last  
 7508 reset. This counter may be reset by writing zero to the attribute.
- 7509 **4.9.2.2.6.18**      ***AverageRMSUnderVoltageCounter***
- 7510 The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last  
 7511 reset. This counter may be reset by writing zero to the attribute.
- 7512 **4.9.2.2.6.19**      ***RMSExtremeOverVoltagePeriod***
- 7513 The duration in seconds used to measure an extreme over voltage condition.
- 7514 **4.9.2.2.6.20**      ***RMSExtremeUnderVoltagePeriod***
- 7515 The duration in seconds used to measure an extreme under voltage condition.
- 7516 **4.9.2.2.6.21**      ***RMSVoltageSagPeriod***
- 7517 The duration in seconds used to measure a voltage sag condition.
- 7518 **4.9.2.2.6.22**      ***RMSVoltageSwellPeriod***
- 7519 The duration in seconds used to measure a voltage swell condition.

7520 **4.9.2.2.7**      **AC Formatting**

7521 **Table 4-36. AC Formatting Attributes**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Default</b>	<b>M/O</b>
0x0600	<i>ACVoltageMultiplier</i>	uint16	0x0001 – 0xffff	RP <sup>66</sup>	0x0001	O
0x0601	<i>ACVoltageDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0602	<i>ACCurrentMultiplier</i>	uint16	0x0001 – 0xffff	RP	0x0001	O

<sup>66</sup> CCB 2236 for all multipliers and divisors that may change, though less frequently than the values

Id	Name	Type	Range	Acc	Default	M/O
0x0603	<i>ACCurrentDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0604	<i>ACPowerMultiplier</i>	uint16	0x0001 – 0xffff	RP	0x0001	O
0x0605	<i>ACPowerDivisor</i>	uint16	0x0001 – 0xffff	RP	0x0001	O

7522 **4.9.2.2.7.1 ACVoltageMultiplier**

7523 Provides a value to be multiplied against the *InstantaneousVoltage* and *RMSVoltage* attributes. This attribute must be  
7524 used in conjunction with the *ACVoltageDivisor* attribute. 0x0000 is an invalid value for this attribute.

7525 **4.9.2.2.7.2 ACVoltageDivisor**

7526 Provides a value to be divided against the *InstantaneousVoltage* and *RMSVoltage* attributes. This attribute must be  
7527 used in conjunction with the *ACVoltageMultiplier* attribute. 0x0000 is an invalid value for this attribute.

7528 **4.9.2.2.7.3 ACCurrentMultiplier**

7529 Provides a value to be multiplied against the *InstantaneousCurrent* and *RMSCurrent* attributes. This attribute must be  
7530 used in conjunction with the *ACCurrentDivisor* attribute. 0x0000 is an invalid value for this attribute.

7531 **4.9.2.2.7.4 ACCurrentDivisor**

7532 Provides a value to be divided against the *ACCurrent*, *InstantaneousCurrent* and *RMSCurrent* attributes. This attribute  
7533 must be used in conjunction with the *ACCurrentMultiplier* attribute. 0x0000 is an invalid value for this attribute.

7534 **4.9.2.2.7.5 ACPowerMultiplier**

7535 Provides a value to be multiplied against the *InstantaneousPower* and *ActivePower* attributes. This attribute must be  
7536 used in conjunction with the *ACPowerDivisor* attribute. 0x0000 is an invalid value for this attribute.

7537 **4.9.2.2.7.6 ACPowerDivisor**

7538 Provides a value to be divided against the *InstantaneousPower* and *ActivePower* attributes. This attribute must be used  
7539 in conjunction with the *ACPowerMultiplier* attribute. 0x0000 is an invalid value for this attribute.

7540 **4.9.2.2.8 DC Manufacturer Threshold Alarms**

7541 **Table 4-37. DC Manufacturer Threshold Alarms Attributes**

Id	Name	Type	Range	Access	Default	M/O
0x0700	<i>DCOverloadAlarmsMask</i>	map8	0000 00xx	RW	0000 0000	O
0x0701	<i>DCVoltageOverload</i>	int16	-32768 – 32767	R	0xffff	O
0x0702	<i>DCCurrentOverload</i>	int16	-32768 – 32767	R	0xffff	O

7542 **4.9.2.2.8.1 DCOverloadAlarmsMask**

7543 Specifies which configurable alarms may be generated, as listed in Figure 4-4. A ‘1’ in each bit position enables the  
7544 alarm.

7545

**Figure 4-4. The DC Overload Alarm Mask**

Bit	Description
Bit0	Voltage Overload
Bit1	Current Overload

7546 **4.9.2.2.8.2 DCVoltageOverload**

7547 Specifies the alarm threshold, set by the manufacturer, for the maximum output voltage supported by device. The  
 7548 value is multiplied and divided by the *DCVoltageMultiplier* the *DCVoltageDivisor* respectively.

7549 **4.9.2.2.8.3 DCCurrentOverload**

7550 Specifies the alarm threshold, set by the manufacturer, for the maximum output current supported by device. The  
 7551 value is multiplied and divided by the *DCCurrentMultiplier* and *DCCurrentDivider* respectively.

7552 **4.9.2.2.9 AC Manufacturer Threshold Alarms**

7553 **Table 4-38. AC Manufacturer Threshold Alarms Attributes**

Id	Name	Type	Range	Access	Default	MO
0x0800	<i>ACAlarmsMask</i>	map16	0000 xxxx	RW	0000 0000	O
0x0801	<i>ACVoltageOverload</i>	int16	-32768 – 32767	R	0xffff	O
0x0802	<i>ACCCurrentOverload</i>	int16	-32768 – 32767	R	0xffff	O
0x0803	<i>ACActivePowerOverload</i>	int16	-32768 – 32767	R	0xffff	O
0x0804	<i>ACReactivePowerOverload</i>	int16	-32768 – 32767	R	0xffff	O
0x0805	<i>AverageRMSOverVoltage</i>	int16	-32768 – 32767	R		O
0x0806	<i>AverageRMSUnderVoltage</i>	int16	-32768 – 32767	R		O
0x0807	<i>RMSExtremeOverVoltage</i>	int16	-32768 – 32767	RW		O
0x0808	<i>RMSExtremeUnderVoltage</i>	int16	-32768 – 32767	RW		O
0x0809	<i>RMSVoltageSag</i>	int16	-32768 – 32767	RW		O
0x080a	<i>RMSVoltageSwell</i>	int16	-32768 – 32767	RW		O

7554 **4.9.2.2.9.1 ACAlarmsMask**

7555 Specifies which configurable alarms may be generated, as listed in Figure 4-5. A ‘1’ in each bit position enables the  
 7556 alarm.

7557

**Figure 4-5. The *ACAlarmsMask* Attribute**

Bit	Description
Bit0	Voltage Overload
Bit1	Current Overload
Bit2	Active Power Overload
Bit3	Reactive Power Overload
Bit4	Average RMS Over Voltage
Bit5	Average RMS Under Voltage
Bit6	RMS Extreme Over Voltage
Bit7	RMS Extreme Under Voltage
Bit8	RMS Voltage Sag
Bit9	RMS Voltage Swell

7558 **4.9.2.2.9.2** *ACVoltageOverload*

7559 Specifies the alarm threshold, set by the manufacturer, for the maximum output voltage supported by device. The  
7560 value is multiplied and divided by the *ACVoltageMultiplier* the *ACVoltageDivisor*, respectively. The value is voltage  
7561 RMS.

7562 **4.9.2.2.9.3** *ACCurrentOverload*

7563 Specifies the alarm threshold, set by the manufacturer, for the maximum output current supported by device. The  
7564 value is multiplied and divided by the *ACCurrentMultiplier* and *ACCurrentDivider*, respectively. The value is current  
7565 RMS.

7566 **4.9.2.2.9.4** *ACActivePowerOverload*

7567 Specifies the alarm threshold, set by the manufacturer, for the maximum output active power supported by device.  
7568 The value is multiplied and divided by the *ACPowerMultiplier* and *ACPowerDivisor*, respectively.

7569 **4.9.2.2.9.5** *ACReactivePowerOverload*

7570 Specifies the alarm threshold, set by the manufacturer, for the maximum output reactive power supported by device.  
7571 The value is multiplied and divided by the *ACPowerMultiplier* and *ACPowerDivisor*, respectively.

7572 **4.9.2.2.9.6** *AverageRMSOverVoltage*

7573 The average RMS voltage above which an over voltage condition is reported. The threshold shall be configurable  
7574 within the specified operating range of the electricity meter. The value is multiplied and divided by the  
7575 *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

7576 **4.9.2.2.9.7** *AverageRMSUnderVoltage*

7577 The average RMS voltage below which an under voltage condition is reported. The threshold shall be configurable  
7578 within the specified operating range of the electricity meter. The value is multiplied and divided by the  
7579 *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

7580 **4.9.2.2.9.8** ***RMSExtremeOverVoltage***

7581 The RMS voltage above which an extreme under voltage condition is reported. The threshold shall be configurable  
 7582 within the specified operating range of the electricity meter. The value is multiplied and divided by the  
 7583 *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

7584 **4.9.2.2.9.9** ***RMSExtremeUnderVoltage***

7585 The RMS voltage below which an extreme under voltage condition is reported. The threshold shall be configurable  
 7586 within the specified operating range of the electricity meter. The value is multiplied and divided by the  
 7587 *ACVoltageMultiplier* and *ACVoltageDivisor*, respectively.

7588 **4.9.2.2.9.10** ***RMSVoltageSag***

7589 The RMS voltage below which a sag condition is reported. The threshold shall be configurable within the specified  
 7590 operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and  
 7591 *ACVoltageDivisor*, respectively.

7592 **4.9.2.2.9.11** ***RMSVoltageSwell***

7593 The RMS voltage above which a swell condition is reported. The threshold shall be configurable within the specified  
 7594 operating range of the electricity meter. The value is multiplied and divided by the *ACVoltageMultiplier* and  
 7595 *ACVoltageDivisor*, respectively.

7596 **4.9.2.2.10** ***AC Phase B Measurements***

7597 **Table 4-39. AC Phase B Measurements Attributes**

Id	Name	Type	Range	Acc	Def	M/O
0x0901	<i>LineCurrentPhB</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0902	<i>ActiveCurrentPhB</i>	int16	-32768 – 32767	RP	0x8000	O
0x0903	<i>ReactiveCurrentPhB</i>	int16	-32768 – 32767	RP	0x8000	O
0x0905	<i>RMSVoltagePhB</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0906	<i>RMSVoltageMinPhB</i>	uint16	0x0000 – 0xffff	R	0x8000	O
0x0907	<i>RMSVoltageMaxPhB</i>	uint16	0x0000 – 0xffff	R	0x8000	O
0x0908	<i>RMSCurrentPhB</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0909	<i>RMSCurrentMinPhB</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x090a	<i>RMSCurrentMaxPhB</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x090b	<i>ActivePowerPhB</i>	int16	-32768 – 32767	RP	0x8000	O
0x090c	<i>ActivePowerMinPhB</i>	int16	-32768 – 32767	R	0x8000	O
0x090d	<i>ActivePowerMaxPhB</i>	int16	-32768 – 32767	R	0x8000	O

Id	Name	Type	Range	Acc	Def	M/O
0x090e	<i>ReactivePowerPhB</i>	int16	-32768 – 32767	RP	0x8000	O
0x090f	<i>ApparentPowerPhB</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0910	<i>PowerFactorPhB</i>	int8	-100 to +100	R	0x00	O
0x0911	<i>AverageRMSVoltageMeasurementPeriodPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0912	<i>AverageRMSOverVoltageCounterPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0913	<i>AverageRMSUnderVoltageCounterPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0914	<i>RMSExtremeOverVoltagePeriodPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0915	<i>RMSExtremeUnderVoltagePeriodPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0916	<i>RMSVoltageSagPeriodPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0917	<i>RMSVoltageSwellPeriodPhB</i>	uint16	0x0000 – 0xffff	RW	0x0000	O

7598 **4.9.2.2.10.1** *LineCurrentPhB*

7599 Represents the Phase B, AC line current (Square root sum of active and reactive currents) value at the moment in time  
7600 the attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000 is returned.

7601 **4.9.2.2.10.2** *ActiveCurrentPhB*

7602 Represents the Phase B, AC active/resistive current value at the moment in time the attribute is read, in Amps (A).  
7603 Positive values indicate power delivered to the premises where negative values indicate power received from the  
7604 premises.

7605 **4.9.2.2.10.3** *ReactiveCurrentPhB*

7606 Represents the Phase B, AC reactive current value at the moment in time the attribute is read, in Amps (A). Positive  
7607 values indicate power delivered to the premises where negative values indicate power received from the premises.

7608 **4.9.2.2.10.4** *RMSVoltagePhB*

7609 Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of  
7610 0xffff is returned.

7611 **4.9.2.2.10.5** *RMSVoltageMinPhB*

7612 Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
7613 0xffff until a measurement is made.

7614 **4.9.2.2.10.6** *RMSVoltageMaxPhB*

7615 Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
7616 0xffff until a measurement is made.

7617 **4.9.2.2.10.7** *RMSCurrentPhB*



- 7618 Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is  
 7619 returned.
- 7620 **4.9.2.2.10.8 *RMSCurrentMinPhB***
- 7621 Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
 7622 0x8000 until a measurement is made.
- 7623 **4.9.2.2.10.9 *RMSCurrentMaxPhB***
- 7624 Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
 7625 0x8000 until a measurement is made.
- 7626 **4.9.2.2.10.10 *ActivePowerPhB***
- 7627 Represents the Phase B, current demand of active power delivered or received at the premises, in Watts (W). Positive  
 7628 values indicate power delivered to the premises where negative values indicate power received from the premises.
- 7629 **4.9.2.2.10.11 *ActivePowerMinPhB***
- 7630 Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
 7631 0x8000 until a measurement is made.
- 7632 **4.9.2.2.10.12 *ActivePowerMaxPhB***
- 7633 Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
 7634 0x8000 until a measurement is made.
- 7635 **4.9.2.2.10.13 *ReactivePowerPhB***
- 7636 Represents the Phase B, current demand of reactive power delivered or received at the premises, in VAr. Positive  
 7637 values indicate power delivered to the premises where negative values indicate power received from the premises.
- 7638 **4.9.2.2.10.14 *ApparentPowerPhB***
- 7639 Represents the Phase B, current demand of apparent (Square root of active and reactive power) power, in VA.
- 7640 **4.9.2.2.10.15 *PowerFactorPhB***
- 7641 Contains the PhaseB, Power Factor ratio in 1/100ths.
- 7642 **4.9.2.2.10.16 *AverageRMSVoltageMeasurementPeriodPhB***
- 7643 The Period in seconds that the RMS voltage is averaged over.
- 7644 **4.9.2.2.10.17 *AverageRMSOverVoltageCounterPhB***
- 7645 The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last  
 7646 reset. This counter may be reset by writing zero to the attribute.
- 7647 **4.9.2.2.10.18 *AverageRMSUnderVoltageCounterPhB***
- 7648 The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last  
 7649 reset. This counter may be reset by writing zero to the attribute.
- 7650 **4.9.2.2.10.19 *RMSExtremeOverVoltagePeriodPhB***
- 7651 The duration in seconds used to measure an extreme over voltage condition.
- 7652 **4.9.2.2.10.20 *RMSExtremeUnderVoltagePeriodPhB***
- 7653 The duration in seconds used to measure an extreme under voltage condition.

7654 **4.9.2.2.10.21** *RMSVoltageSagPeriodPhB*

7655 The duration in seconds used to measure a voltage sag condition.

7656 **4.9.2.2.10.22** *RMSVoltageSwellPeriodPhB*

7657 The duration in seconds used to measure a voltage swell condition.

7658 **4.9.2.2.11** **AC Phase C Measurements**

7659 **Table 4-40. AC Phase C Measurements Attributes**

Id	Name	Type	Range	Acc	Def	M/O
0x0a01	<i>LineCurrentPhC</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a02	<i>ActiveCurrentPhC</i>	int16	-32768 – 32767	RP	0x8000	O
0x0a03	<i>ReactiveCurrentPhC</i>	int16	-32768 – 32767	RP	0x8000	O
0x0a05	<i>RMSVoltagePhC</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a06	<i>RMSVoltageMinPhC</i>	uint16	0x0000 – 0xffff	R	0x8000	O
0x0a07	<i>RMSVoltageMaxPhC</i>	uint16	0x0000 – 0xffff	R	0x8000	O
0x0a08	<i>RMSCurrentPhC</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a09	<i>RMSCurrentMinPhC</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0a0a	<i>RMSCurrentMaxPhC</i>	uint16	0x0000 – 0xffff	R	0xffff	O
0x0a0b	<i>ActivePowerPhC</i>	int16	-32768 – 32767	RP	0x8000	O
0x0a0c	<i>ActivePowerMinPhC</i>	int16	-32768 – 32767	R	0x8000	O
0x0a0d	<i>ActivePowerMaxPhC</i>	int16	-32768 – 32767	R	0x8000	O
0x0a0e	<i>ReactivePowerPhC</i>	int16	-32768 – 32767	RP	0x8000	O
0x0a0f	<i>ApparentPowerPhC</i>	uint16	0x0000 – 0xffff	RP	0xffff	O
0x0a10	<i>PowerFactorPhC</i>	int8	-100 to +100	R	0x00	O
0x0a11	<i>AverageRMSVoltageMeasurementPeriodPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a12	<i>AverageRMSOverVoltageCounterPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a13	<i>AverageRMSUnderVoltageCounterPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a14	<i>RMSExtremeOverVoltagePeriodPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a15	<i>RMSExtremeUnderVoltagePeriodPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O

Id	Name	Type	Range	Acc	Def	M/O
0x0a16	<i>RMSVoltageSagPeriodPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0a17	<i>RMSVoltageSwellPeriodPhC</i>	uint16	0x0000 – 0xffff	RW	0x0000	O

7660 **4.9.2.2.11.1** *LineCurrentPhC*

7661 Represents the Phase C, AC line current (Square root of active and reactive current) value at the moment in time the  
 7662 attribute is read, in Amps (A). If the instantaneous current cannot be measured, a value of 0x8000 is returned.

7663 **4.9.2.2.11.2** *ActiveCurrentPhC*

7664 Represents the Phase C, AC active/resistive current value at the moment in time the attribute is read, in Amps (A).  
 7665 Positive values indicate power delivered to the premises where negative values indicate power received from the  
 7666 premises.

7667 **4.9.2.2.11.3** *ReactiveCurrentPhC*

7668 Represents the Phase C, AC reactive current value at the moment in time the attribute is read, in Amps (A). Positive  
 7669 values indicate power delivered to the premises where negative values indicate power received from the premises.

7670 **4.9.2.2.11.4** *RMSVoltagePhC*

7671 Represents the most recent RMS voltage reading in Volts (V). If the RMS voltage cannot be measured, a value of  
 7672 0xffff is returned.

7673 **4.9.2.2.11.5** *RMSVoltageMinPhC*

7674 Represents the lowest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
 7675 0xffff until a measurement is made.

7676 **4.9.2.2.11.6** *RMSVoltageMaxPhC*

7677 Represents the highest RMS voltage value measured in Volts (V). After resetting, this attribute will return a value of  
 7678 0xffff until a measurement is made.

7679 **4.9.2.2.11.7** *RMSCurrentPhC*

7680 Represents the most recent RMS current reading in Amps (A). If the power cannot be measured, a value of 0xffff is  
 7681 returned.

7682 **4.9.2.2.11.8** *RMSCurrentMinPhC*

7683 Represents the lowest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
 7684 0x8000 until a measurement is made.

7685 **4.9.2.2.11.9** *RMSCurrentMaxPhC*

7686 Represents the highest RMS current value measured in Amps (A). After resetting, this attribute will return a value of  
 7687 0x8000 until a measurement is made.

7688 **4.9.2.2.11.10** *ActivePowerPhC*

7689 Represents the Phase C, current demand of active power delivered or received at the premises, in Watts (W). Positive  
 7690 values indicate power delivered to the premises where negative values indicate power received from the premises.

7691 **4.9.2.2.11.11** *ActivePowerMinPhC*

7692 Represents the lowest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
7693 0x8000 until a measurement is made.

7694 **4.9.2.2.11.12    *ActivePowerMaxPhC***

7695 Represents the highest AC power value measured in Watts (W). After resetting, this attribute will return a value of  
7696 0x8000 until a measurement is made.

7697 **4.9.2.2.11.13    *ReactivePowerPhC***

7698 Represents the Phase C, current demand of reactive power delivered or received at the premises, in VAR. Positive  
7699 values indicate power delivered to the premises where negative values indicate power received from the premises.

7700 **4.9.2.2.11.14    *ApparentPowerPhC***

7701 Represents the Phase C, current demand of apparent (Square root of active and reactive power) power, in VA.

7702 **4.9.2.2.11.15    *PowerFactorPhC***

7703 Contains the Phase C, Power Factor ratio in 1/100ths.

7704 **4.9.2.2.11.16    *AverageRMSVoltageMeasurementPeriodPhC***

7705 The Period in seconds that the RMS voltage is averaged over

7706 **4.9.2.2.11.17    *AverageRMSOverVoltageCounterPhC***

7707 The number of times the average RMS voltage, has been above the *AverageRMS OverVoltage* threshold since last  
7708 reset. This counter may be reset by writing zero to the attribute.

7709 **4.9.2.2.11.18    *AverageRMSUnderVoltageCounterPhC***

7710 The number of times the average RMS voltage, has been below the *AverageRMS underVoltage* threshold since last  
7711 reset. This counter may be reset by writing zero to the attribute.

7712 **4.9.2.2.11.19    *RMSExtremeOverVoltagePeriodPhC***

7713 The duration in seconds used to measure an extreme over voltage condition.

7714 **4.9.2.2.11.20    *RMSExtremeUnderVoltagePeriodPhC***

7715 The duration in seconds used to measure an extreme under voltage condition.

7716 **4.9.2.2.11.21    *RMSVoltageSagPeriodPhC***

7717 The duration in seconds used to measure a voltage sag condition.

7718 **4.9.2.2.11.22    *RMSVoltageSwellPeriodPhC***

7719 The duration in seconds used to measure a voltage swell condition.

7720 **4.9.2.3    Server Commands**

7721 **4.9.2.3.1    Commands Generated**

7722 The command IDs generated by the electrical measurement server cluster are listed in Table 4-41.

7723

**Table 4-41. Generated Command ID's for the Electrical Measurement Server**

Command Identifier	Description	M/O
0x00	Get Profile Info Response Command	O
0x01	Get Measurement Profile Response Command	O

7724 **4.9.2.3.1.1 Get Profile Info Response Command**

7725 The Get Profile Info Response Command shall be formatted as illustrated in Figure 4-6.

7726 **Figure 4-6. Format of the Get Profile Info Response Command**

Octets	1	1	1	Variable
Data Type	uint8	enum8	uint8	Array of attribute IDs (two-byte unsigned values)
Field Name	Profile Count	ProfileIntervalPeriod	MaxNumberOfIntervals	ListOfAttributes

7727 **4.9.2.3.1.2 Payload Details**

7728 **Profile Count:** Total number of supported profile.

7729 **ProfileIntervalPeriod:** Represents the interval or time frame used to capture parameter for profiling purposes.  
 7730 ProfileIntervalPeriod is an enumerated field representing the timeframes listed in Figure 4-7.

7731 **Figure 4-7. ProfileIntervalPeriod**

Enumerated Value	Time Frame
0	Daily
1	60 minutes
2	30 minutes
3	15 minutes
4	10 minutes
5	7.5 minutes
6	5 minutes
7	2.5 minutes

7732 **MaxNumberOfIntervals:** Represents the maximum number of intervals the device is capable of returning in one  
 7733 Get Measurement Profile Response command. It is required MaxNumberOfIntervals fit within the default  
 7734 Fragmentation ASDU size of 128 bytes, or an optionally agreed upon larger Fragmentation ASDU size supported by  
 7735 both devices as per the application profile supported by the devices.

7736 **ListOfAttributes:** Represents the list of attributes being profiled.

7737 **4.9.2.3.1.3 When Generated**

7738 This command is generated when the Client command GetProfileInfo is received.

7739 **4.9.2.3.1.4 Get Measurement Profile Response Command**

7740 The Get Measurement Profile Response Command shall be formatted as illustrated in Figure 4-8.

7741 **Figure 4-8. Format of the Get Measurement Profile Response Command**

<b>Octets</b>	4	1	1	1	1	Variable
<b>Data Type</b>	UTC	enum8	enum8	uint8	attribId	Array of Attribute values
<b>Field Name</b>	StartTime	Status	ProfileIntervalPeriod	NumberOfIntervalsDelivered	Attribute Id	Intervals

7742 **4.9.2.3.1.5 Payload Details**

7743 **StartTime:** 32-bit value (in UTC) representing the end time of the most chronologically recent interval being requested. Example: Data collected from 2:00 PM to 3:00 PM would be specified as a 3:00 PM interval (end time).  
7744

7745 **Status:** Table status enumeration in Table 4-42 lists the valid values returned in the Status field.

7746 **Table 4-42. List of Status Valid Values**

Status Value	Description
0x00	Success
0x01	Attribute Profile not supported
0x02	Invalid Start Time
0x03	More intervals requested than can be returned
0x04	No intervals available for the requested time

7747  
7748 **ProfileIntervalPeriod:** Represents the interval or time frame used to capture parameter for profiling purposes. Refer to table “ProfileIntervalPeriod”.  
7749

7750 **NumberOfIntervalsDelivered:** Represents the number of intervals the device is returning. Please note the number of intervals returned in the Get Measurement Profile Response command can be calculated when the packets are received  
7751 and can replace the usage of this field. The intent is to provide this information as a convenience.  
7752

7753 **AttributeID:** The attribute that has been profiled by the application.

7754 **Intervals:** Series of interval data captured using the period specified by the ProfileIntervalPeriod field. The content  
 7755 of the interval data depend of the type of information requested using the **AttributeID** field in the Get Measurement  
 7756 Profile Command. Data is organized in a reverse chronological order, the oldest intervals are transmitted first and the  
 7757 newest interval is transmitted last. Invalid intervals should be marked as 0xffff. For scaling and data type use the  
 7758 respective attribute set as defined above in attribute sets.

7759 **4.9.2.3.1.6 When Generated**

7760 This command is generated when the Client command GetMeasurementProfile is received.

7761 **4.9.2.4 Client Commands**

7762 **4.9.2.4.1 Commands Generated**

7763 The command ID’s generated by the electrical measurement client cluster are listed in Table 4-43.

7764 **Table 4-43. Generated Command IDs for the Electrical Measurement Client**

Command Identifier	Description	M/O
0x00	Get Profile Info Command	O
0x01	Get Measurement Profile Command	O

7765 **4.9.2.4.1.1 Get Profile Info Command**

7766 This command has no payload.

7767 **4.9.2.4.1.2 Effect on Receipt**

7768 On receipt of this command, the device shall send a Get Profile Info Response Command. A ZCL default response  
 7769 with status UNSUP\_CLUSTER\_COMMAND shall be returned if command is not supported on the device.

7770 **4.9.2.4.1.3 Get Measurement Profile Command**

7771 The Get Measurement Profile Command shall be formatted as illustrated in Figure 4-9.

7772 **Figure 4-9. Format of the Get Measurement Profile Command**

<b>Octets</b>	2	1	1
<b>Data Type</b>	attribId	UTC Time	uint8
<b>Field Name</b>	Attribute ID	Start Time	NumberOfIntervals

7773 **4.9.2.4.1.4 Payload Details**

7774 **Attribute ID:** The electricity measurement attribute being profiled.

7775 **StartTime:** 32-bit value (in UTCtime) used to select an Intervals block from all the Intervals blocks available. The  
 7776 Intervals block returned is the most recent block with its StartTime equal or greater to the one provided.

7777 **NumberOfIntervals:** Represents the number of intervals being requested. This value can't exceed the size stipulated  
7778 in the MaxNumberOfIntervals field of Get Profile Info Response Command. If more intervals are requested than can  
7779 be delivered, the GetMeasurementProfileResponse will return the number of intervals equal to  
7780 MaxNumberOfIntervals. If fewer intervals available for the time period then only those available are returned.

7781 **4.9.2.4.1.5 Effect on Receipt**

7782 On receipt of this command, the device shall send a Get Measurement Profile Response Command. A ZCL default  
7783 response with status UNSUP\_CLUSTER\_COMMAND shall be returned if command is not supported on the device.

7784 **4.10 Electrical Conductivity Measurement**

7785 **4.10.1 Overview**

7786 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
7787 etc.

7788 The server cluster provides an interface to Electrical Conductivity measurement functionality.

7789 **4.10.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

7790 **4.10.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	EC	Type 2 (server to client)

7791 **4.10.1.3 Cluster Identifiers**

Identifier	Name
0x040A	Electrical Conductivity

7792 **4.10.2 Server**

7793 **4.10.2.1 Attributes**

7794 **Table 4-44. Attributes of the Electrical Conductivity Measurement server cluster**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	uint16	<i>MinMeasuredValue</i> to <i>MaxMeasuredValue</i>	RP	0xffff	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0000 to <i>MaxMeasuredValue</i> -1	R	0xffff	M
0x0002	<i>MaxMeasuredValue</i>	uint16	<i>MinMeasuredValue</i> +1 to 0xfffe	R	0xffff	M



Id	Name	Type	Range	Acc	Def	MO
0x0003	<i>Tolerance</i>	uint16	0x0000 to 0x0064	R		O

7795 **4.10.2.1.1 *MeasuredValue* Attribute**

7796 *MeasuredValue* represents the Electrical Conductivity in EC or mS/m (milli-Siemens per meter) as follows:

7797 *MeasuredValue* = 10 x Electrical Conductivity in mS/m. The maximum resolution this format allows is 0.1.

7798 A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

7799 *MeasuredValue* is updated continuously as new measurements are made.

7800 **4.10.2.1.2 *MinMeasuredValue* Attribute**

7801 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured.

7802 A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

7803 **4.10.2.1.3 *MaxMeasuredValue* Attribute**

7804 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured.

7805 A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

7806 **4.10.2.1.4 *Tolerance* Attribute**

7807 See 4.1.3.

7808 **4.10.2.2 Commands**

7809 No cluster specific commands are generated or received by the server cluster.

7810 **4.10.3 Client**

7811 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7812

7813 **4.11 pH Measurement**

---

7814 **4.11.1 Overview**

7815 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7816 etc.

7817 The server cluster provides an interface to pH measurement functionality.

7818 **4.11.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

7819 **4.11.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PH	Type 2 (server to client)

7820 **4.11.1.3 Cluster Identifiers**

Identifier	Name
0x0409	pH Measurement

7821 **4.11.2 Server**

7822 **4.11.2.1 Attributes**

7823 **Table 4-45. Attributes of the pH Measurement server cluster**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>MeasuredValue</i>	uint16	<i>MinMeasuredValue</i> to <i>MaxMeasuredValue</i>	RP	0xffff	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0000 to <i>MaxMeasuredValue</i> -1	R	0xffff	M
0x0002	<i>MaxMeasuredValue</i>	uint16	<i>MinMeasuredValue</i> +1 to 0x0578	R	0xffff	M
0x0003	<i>Tolerance</i>	uint16	0x0000 to 0x00c8	R		O

7824 **4.11.2.1.1 *MeasuredValue* Attribute**

7825 *MeasuredValue* represents the pH with no units as follows:  $MeasuredValue = 100 \times pH$ .

7826 Where  $0.00 \leq pH \leq 14.00$ , corresponding to a *MeasuredValue* in the range 0x0000 to 0x0578. The maximum  
7827 resolution this format allows is 0.01, this is to accommodate certain applications where such resolution is necessary.

7828 A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

7829 *MeasuredValue* is updated continuously as new measurements are made.

7830 **4.11.2.1.2 *MinMeasuredValue* Attribute**

7831 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured.  
7832 A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

7833 **4.11.2.1.3 *MaxMeasuredValue* Attribute**

7834 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured.  
7835 A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

7836 **4.11.2.1.4 *Tolerance* Attribute**

7837 See 4.1.3.

7838 **4.11.2.2 Commands**

7839 No cluster specific commands are generated or received by the server cluster.

7840 **4.11.3 Client**

7841 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7842 **4.12 Wind Speed Measurement**

7843 **4.12.1 Overview**

7844 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 7845 etc.

7846 The server cluster provides an interface to Wind Speed measurement functionality.

7847 **4.12.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

7848 **4.12.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	WSPD	Type 2 (server to client)

7849 **4.12.1.3 Cluster Identifiers**

Identifier	Name
0x040b	Wind Speed Measurement

7850 **4.12.2 Server**

7851 **4.12.2.1 Attributes**

7852 **Table 4-46. Attributes of the Wind Speed Measurement server cluster**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>MeasuredValue</i>	uint16	<i>MinMeasuredValue</i> to <i>MaxMeasuredValue</i>	RP	0xffff	M
0x0001	<i>MinMeasuredValue</i>	uint16	0x0000 to <i>MaxMeasuredValue</i> -1	R	0xffff	M
0x0002	<i>MaxMeasuredValue</i>	uint16	<i>MinMeasuredValue</i> +1 to 0xfffe	R	0xffff	M

Id	Name	Type	Range	Acc	Def	M/O
0x0003	<i>Tolerance</i>	uint16	0x0000 to 0x0308	R		O

7853

7854 **4.12.2.1.1 *MeasuredValue* Attribute**

7855 *MeasuredValue* represents the Wind Speed in m/s (meters per second) as follows:

7856  $MeasuredValue = 100 \times \text{Wind Speed in m/s}$ . The maximum resolution this format allows is 0.01.

7857 A *MeasuredValue* of 0xffff SHALL indicate an unknown value, otherwise the range SHALL be as described in 4.1.3.

7858 *MeasuredValue* is updated continuously as new measurements are made.

7859 **4.12.2.1.2 *MinMeasuredValue* Attribute**

7860 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured.

7861 A *MinMeasuredValue* of 0xffff indicates that the minimum value is not defined. See 4.1.3 for more details.

7862 **4.12.2.1.3 *MaxMeasuredValue* Attribute**

7863 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured.

7864 A *MaxMeasuredValue* of 0xffff indicates that the maximum value is not defined. See 4.1.3 for more details.

7865 **4.12.2.1.4 *Tolerance* Attribute**

7866 See 4.1.3.

7867 **4.12.2.2 Commands**

7868 No cluster specific commands are generated or received by the server cluster.

7869 **4.12.3 Client**

7870 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.

7871 **4.13 Concentration Measurement**

---

7872 **4.13.1 Overview**

7873 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

7875 The server cluster provides an interface to concentration measurement functionality. The measurement is reportable and may be configured for reporting. Concentration measurements include, but are not limited to, levels in gases, such as CO, CO2, and ethylene, or in fluids and solids, such as dissolved oxygen, chemicals & pesticides.

7878 **4.13.1.1 Revision History**

Rev	Description
-----	-------------

1	mandatory global <i>ClusterRevision</i> attribute added
---	---

7879 **4.13.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CONC	Type 2 (server to client)

7880 **4.13.1.3 Cluster Identifiers**

7881 The table below is a list of Cluster Ids that conform to this specification. More than one ambient substance may be  
 7882 supported by the same Cluster Id (e.g. Water and Alcohol). This would make the Cluster Id generic to the ambient  
 7883 substance. A new Cluster Id may also be added that is limited to a single ambient substance to provide more specific  
 7884 self-description. If both generic and specific Cluster Ids appear on an endpoint, then a single instance of the cluster  
 7885 exists on the endpoint, and either Cluster Id can be used to access the cluster interface.

Cluster Id	Substance Measured	Ambient Substance	Units	Notes (not normative or prescribed)
0x040c	Carbon Monoxide (CO)	Air	Volume	
0x040d	Carbon Dioxide (CO2)	Air	Volume	
0x040e	Ethylene (CH2)	Air	Volume	
0x040f	Ethylene Oxide (C2H4O)	Air	Volume	
0x0410	Hydrogen (H)	Air	Volume	
0x0411	Hydrogen Sulfide (H2S)	Air	Volume	
0x0412	Nitric Oxide (NO)	Air	Volume	
0x0413	Nitrogen Dioxide (NO2)	Air	Volume	
0x0414	Oxygen (O2)	Air	Volume	
0x0415	Ozone (O3)	Air	Volume	
0x0416	Sulfur Dioxide (SO2)	Air	Volume	
0x0417	Dissolved Oxygen (DO)	Water	Mass	
0x0418	Bromate	Drinking Water	Volume	typical range example: not detected to 3.6 PPB typical value example: 1.79 PPB
0x0419	Chloramines	Drinking Water	Volume	typical range example: 0.9 to 3.8 PPM typical value example: 2.87 PPM
0x041a	Chlorine	Drinking Water	Volume	typical range example: 0.1 to 2.4 PPM typical value example: 1.28 PPM

0x041b	Fecal coliform & E. Coli	Drinking Water	Volume	Percent of positive samples typical value example: 0
0x041b	Fluoride	Drinking Water	Volume	typical range example: 0 to 100 PPM typical value example: 0.72 PPM
0x041d	Haloacetic Acids	Drinking Water	Volume	typical range example: Not Detected to 20 PPB typical value example: 14 PPB
0x041e	Total Trihalomethanes	Drinking Water	Volume	typical range example: 0 to 100 PPB typical value example: 44 PPB
0x041f	Total Coliform Bacteria	Drinking Water	Volume	Percent of positive samples typical range example: 0 to 100% typical value example: 1.33%
0x0420	Turbidity	Drinking Water	Volume	Cloudiness of particles in water where an average person would notice a 5 or higher typical range example: 0 to 10 typical value example: 0.18
0x0421	Copper	Drinking Water	Volume	typical range example: 0 to 10 PPM typical value example: 0.191 PPM
0x0422	Lead	Drinking Water	Volume	typical range example: 0 to 10 PPB typical value example: 3.2 PPB
0x0423	Manganese	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 31PPB
0x0424	Sulfate	Drinking Water	Volume	typical range example: 0 to 1000 PPM typical value example: 36 PPM
0x0425	Bromodichloromethane	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 9.6 PPB
0x0426	Bromoform	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 1.1 PPB
0x0427	Chlorodibromomethane	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 6.4 PPB
0x0428	Chloroform	Drinking Water	Volume	typical range example: 0 to 1000 PPB typical value example: 8.0 PPB
0x0429	Sodium	Drinking Water	Volume	typical range example: 0 to 1000 PPM typical value example: 27 PPM
0x042A	PM2.5	Air	Volume	Particulate Matter 2.5 microns or less
0x042B	Formaldehyde	Air	Volume	

7886 **4.13.2 Server**

7887 **4.13.2.1 Attributes**

7888 **Table 4-47. Attributes of the Concentration Measurement server cluster**

Id	Name	Type	Range	Acc	Def	MO
0x0000	<i>MeasuredValue</i>	single	<i>MinMeasuredValue</i> to <i>MaxMeasuredValue</i>	RP	NaN*	M
0x0001	<i>MinMeasuredValue</i>	single	$0 \leq \text{value} < \text{MaxMeasuredValue}$	R	NaN*	M
0x0002	<i>MaxMeasuredValue</i>	single	$\text{MinMeasuredValue} < \text{value} \leq 1$	R	NaN*	M
0x0003	<i>Tolerance</i>	single	<i>MS</i>	R	<i>MS</i>	O

7889 \* see Not a Number: Chapter 2 for data type default and invalid values

7890 **4.13.2.1.1 *MeasuredValue* Attribute**

7891 *MeasuredValue* represents the concentration as a fraction of 1 (one).

7892 A value of NaN indicates that the concentration measurement is unknown or outside the valid range.

7893 *MinMeasuredValue* and *MaxMeasuredValue* define the valid range for *MeasuredValue*.

7894 *MeasuredValue* is updated continuously as new measurements are made.

7895 **4.13.2.1.2 *MinMeasuredValue* Attribute**

7896 The *MinMeasuredValue* attribute indicates the minimum value of *MeasuredValue* that is capable of being measured.  
 7897 A *MinMeasuredValue* of NaN indicates that the *MinMeasuredValue* is not defined. See 4.1.3 for more details.

7898 **4.13.2.1.3 *MaxMeasuredValue* Attribute**

7899 The *MaxMeasuredValue* attribute indicates the maximum value of *MeasuredValue* that is capable of being measured.  
 7900 A *MaxMeasuredValue* of NaN indicates that the *MaxMeasuredValue* is not defined. See 4.1.3 for more details.

7901 **4.13.2.1.4 *Tolerance* Attribute**

7902 See 4.1.3.

7903 **4.13.2.2 Commands**

7904 No cluster specific commands are generated or received by the server cluster.

7905 **4.13.3 Client**

7906 The client cluster has no dependencies, cluster specific attributes nor specific commands generated or received.





7907

# CHAPTER 5 LIGHTING

7908 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 7909 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 7910 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 7911 using [*Rn*] notation.

7912

## 5.1 General Description

7913

### 5.1.1 Introduction

7914 The clusters specified in this document are for use typically in lighting applications, but MAY be used in any  
 7915 application domain.

7916

### 5.1.2 Terms

7917 **Ballast Factor:** A measure of the light output (lumens) of a ballast and lamp combination in comparison to an ANSI  
 7918 standard ballast operated with the same lamp. Multiply the ballast factor by the rated lumens of the lamp to get the  
 7919 light output of the lamp/ballast combination.

7920 **HSV:** Hue, Saturation, Value. A color space, also known as HSB (Hue, Saturation, Brightness). This is a well-known  
 7921 transformation of the RGB (Red, Green, Blue) color space. For more information see e.g.,  
 7922 [http://en.wikipedia.org/wiki/HSV\\_color\\_space](http://en.wikipedia.org/wiki/HSV_color_space).

7923 **Illuminance:** The density of incident luminous flux on a surface. Illuminance is the standard metric for lighting levels,  
 7924 and is measured in lux (lx).

7925

### 5.1.3 Cluster List

7926 This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of  
 7927 clarification. The clusters specified in this document are listed in Table 5-1.

7928

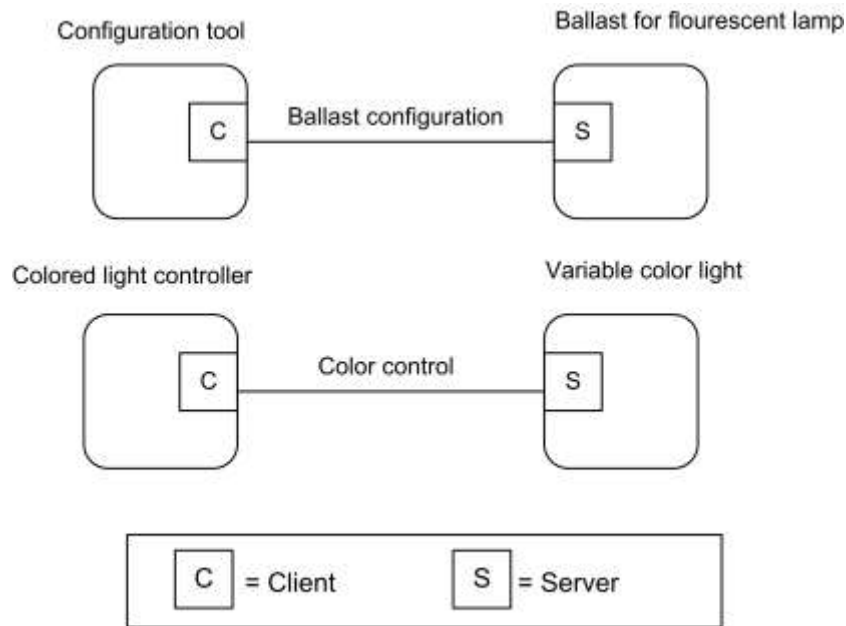
**Table 5-1. Clusters Specified for the Lighting Functional Domain**

ID	Cluster Name	Description
0x0300	Color Control	Attributes and commands for controlling the color of a color-capable light.
0x0301	Ballast Configuration	Attributes and commands for configuring a lighting ballast

7929

7930

**Figure 5-1. Typical Usage of Ballast Configuration and Color Control Clusters**



7931

*Note: Device names are examples for illustration purposes only*

7932

## 5.2 Color Control Cluster

7933

### 5.2.1 Overview

7934

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

7935

7936

This cluster provides an interface for changing the color of a light. Color is specified according to the Commission Internationale de l'Éclairage (CIE) specification CIE 1931 Color Space, [I1]. Color control is carried out in terms of x,y values, as defined by this specification.

7937

7938

7939

Additionally, color MAY optionally be controlled in terms of color temperature, or as hue and saturation values based on optionally variable RGB and W color points. It is recommended that the hue and saturation are interpreted according to the HSV (aka HSB) color model.

7940

7941

7942

Control over luminance is not included, as this is provided by means of the Level Control for Lighting cluster of the General library (see Chapter 3). It is recommended that the level provided by this cluster be interpreted as representing a proportion of the maximum intensity achievable at the current color.

7943

7944

7945

#### 5.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCB 2028
2	added <i>Options</i> attribute, CCB 2085 2104 2124 2230; ZLO 1.0

7946 **5.2.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CC	Type 1 (client to server)

7947 **5.2.1.3 Cluster Identifiers**

Identifier	Name
0x0300	Color Control

7948 **5.2.2 Server**

7949 **5.2.2.1 Dependencies**

7950 **5.2.2.1.1 Coupling color temperature to Level Control<sup>67</sup>**

7951 If the *Level Control for Lighting* cluster identifier 0x0008 is supported on the same endpoint as the *Color Control*  
 7952 cluster and color temperature is supported, it is possible to couple changes in the current level to the color temperature.

7953 The *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster indicates whether the color  
 7954 temperature is to be linked with the *CurrentLevel* attribute in the *Level Control* cluster.

7955 If the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is equal to 1 and the  
 7956 *ColorMode* or *EnhancedColorMode* attribute is set to 0x02 (*color temperature*) then a change in the *CurrentLevel*  
 7957 attribute SHALL affect the *ColorTemperatureMireds* attribute. This relationship is manufacturer specific, with the  
 7958 qualification that the maximum value of the *CurrentLevel* attribute SHALL correspond to a *ColorTemperatureMired*  
 7959 attribute value equal to the *CoupleColorTempToLevelMinMireds* attribute. This relationship is one-way so a change  
 7960 to the *ColorTemperatureMireds* attribute SHALL NOT have any effect on the *CurrentLevel* attribute.

7961 In order to simulate the behavior of an incandescent bulb, a low value of the *CurrentLevel* attribute SHALL be  
 7962 associated with a high value of the *ColorTemperatureMireds* attribute (i.e., a low value of color temperature in  
 7963 kelvins).

7964 If the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is equal to 0, there SHALL  
 7965 be no link between color temperature and current level.

7966 **5.2.2.2 Attributes**

7967 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 7968 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 7969 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 7970 are listed in Table 5-2.

7971 **Table 5-2. Hue Control Attribute Sets**

Attribute Set Identifier	Description
0x000, 0x400	Color Information

<sup>67</sup> ZLO 1.0

0x001	Defined Primaries Information
0x002	Additional Defined Primaries Information
0x003	Defined Color Point Settings

7972 **5.2.2.2.1 Color Information Attribute Set**

7973 The Color Information attribute set contains the attributes summarized below.

7974 **Table 5-3. Attributes of the Color Information Attribute Set**

7975

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0000	CurrentHue	uint8	0x00 – 0xfe	RP	0x00	M <sup>0</sup>
0x0001	CurrentSaturation	uint8	0x00 – 0xfe	RPS	0x00	M <sup>0</sup>
0x0002	RemainingTime	uint16	0x0000 – 0xffff	R	0x00	O
0x0003	CurrentX	uint16	0x0000 - 0xfeff	RP Scene	0x616b (0.381)	M <sup>3</sup>
0x0004	CurrentY	uint16	0x0000 - 0xfeff	RPS	0x607d (0.377)	M <sup>3</sup>
0x0005	DriftCompensation	enum8	0x00 – 0x04	R	-	O
0x0006	CompensationText	string	0 to 254 chars	R	-	O
0x0007	ColorTemperatureMireds	uint16	0x0000 - 0xfeff	RPS <sup>68</sup>	0x00fa (4000K)	M <sup>4</sup>
0x0008	ColorMode	enum8	0x00 – 0x02	R	0x01	M
0x000f	Options <sup>69</sup>	map8		RW	0x00	M
0x4000	<i>EnhancedCurrentHue</i>	uint16	0x0000 – 0xffff	RS	0x0000	M <sup>1</sup>
0x4001	<i>EnhancedColorMode</i>	enum8	0x00 – 0xff	R	0x01 <sup>70</sup>	M
0x4002	<i>ColorLoopActive</i>	uint8	0x00 – 0xff	RS	0x00	M <sup>2</sup>
0x4003	<i>ColorLoopDirection</i>	uint8	0x00 – 0xff	RS	0x00	M <sup>2</sup>
0x4004	<i>ColorLoopTime</i>	uint16	0x0000 – 0xffff	RS	0x0019	M <sup>2</sup>
0x4005	<i>ColorLoopStartEnhancedHue</i>	uint16	0x0000 – 0xffff	R	0x2300	M <sup>2</sup>
0x4006	<i>ColorLoopStoredEnhancedHue</i>	uint16	0x0000 – 0xffff	R	0x0000	M <sup>2</sup>
0x400a	<i>ColorCapabilities</i>	map16	0x0000 – 0x001f	R	0x0000	M
0x400b	<i>ColorTempPhysicalMinMireds</i>	uint16	0x0000 – 0xfeff	R	0x0000	M <sup>4</sup>
0x400c	<i>ColorTempPhysicalMaxMireds</i>	uint16	0x0000 – 0xfeff	R	0xfeff	M <sup>4</sup>
0x400d	<i>CoupleColorTempToLevelMinMireds</i> <sup>71</sup>	uint16	<i>ColorTempPhysicalMinMireds to ColorTempPhysicalMaxMireds</i>	R	MS	M <sup>4*</sup>
0x4010	<i>StartUpColorTemperatureMireds</i> <sup>72</sup>	uint16	0x0000-0xffff	RW	MS	M <sup>4*</sup>

<sup>68</sup> ZLO 1.0

<sup>69</sup> CCB 2085

<sup>70</sup> CCB 2124

<sup>71</sup> ZLO 1.0

<sup>72</sup> ZLO 1.0

7976  $M^i$  = Mandatory if bit  $i$  of the *ColorCapabilities* attribute is equal to 1, otherwise optional.

7977 \* Mandatory if *ColorTemperatureMireds* is supported.

#### 7978 **5.2.2.2.1.1 CurrentHue Attribute**

7979 The *CurrentHue* attribute contains the current hue value of the light. It is updated as fast as practical during commands  
7980 that change the hue.

7981 The hue in degrees SHALL be related to the *CurrentHue* attribute by the relationship  
7982  $Hue = CurrentHue \times 360 / 254$  (*CurrentHue* in the range 0 - 254 inclusive)

7983 If this attribute is implemented then the *CurrentSaturation* and *ColorMode* attributes SHALL also be implemented.

#### 7984 **5.2.2.2.1.2 CurrentSaturation Attribute**

7985 The *CurrentSaturation* attribute holds the current saturation value of the light. It is updated as fast as practical during  
7986 commands that change the saturation.

7987 The saturation SHALL be related to the *CurrentSaturation* attribute by the relationship  
7988  $Saturation = CurrentSaturation / 254$  (*CurrentSaturation* in the range 0 - 254 inclusive)

7989 If this attribute is implemented then the *CurrentHue* and *ColorMode* attributes SHALL also be implemented.

#### 7990 **5.2.2.2.1.3 RemainingTime Attribute**

7991 The *RemainingTime* attribute holds the time remaining, in 1/10ths of a second, until the currently active command  
7992 will be complete.

#### 7993 **5.2.2.2.1.4 CurrentX Attribute**

7994 The *CurrentX* attribute contains the current value of the normalized chromaticity value  $x$ , as defined in the CIE xyY  
7995 Color Space. It is updated as fast as practical during commands that change the color.

7996 The value of  $x$  SHALL be related to the *CurrentX* attribute by the relationship

7997  $x = CurrentX / 65536$  (*CurrentX* in the range 0 to 65279 inclusive)

#### 7998 **5.2.2.2.1.5 CurrentY Attribute**

7999 The *CurrentY* attribute contains the current value of the normalized chromaticity value  $y$ , as defined in the CIE xyY  
8000 Color Space. It is updated as fast as practical during commands that change the color.

8001 The value of  $y$  SHALL be related to the *CurrentY* attribute by the relationship

8002  $y = CurrentY / 65536$  (*CurrentY* in the range 0 to 65279 inclusive)

#### 8003 **5.2.2.2.1.6 DriftCompensation Attribute**

8004 The *DriftCompensation* attribute indicates what mechanism, if any, is in use for compensation for color/intensity drift  
8005 over time. It SHALL be one of the non-reserved values in Table 5-4.

8006 **Table 5-4. Values of the *DriftCompensation* Attribute**

Attribute Value	Description
0x00	None
0x01	Other / Unknown
0x02	Temperature monitoring

0x03	Optical luminance monitoring and feedback
0x04	Optical color monitoring and feedback

8007 **5.2.2.2.1.7 CompensationText Attribute**

8008 The *CompensationText* attribute holds a textual indication of what mechanism, if any, is in use to compensate for  
 8009 color/intensity drift over time.

8010 **5.2.2.2.1.8 ColorTemperatureMireds Attribute**

8011 The *ColorTemperatureMireds* attribute contains a scaled inverse of the current value of the color temperature. The  
 8012 unit of *ColorTemperatureMireds* is the mired (micro reciprocal degree), AKA mirek (micro reciprocal kelvin). It is  
 8013 updated as fast as practical during commands that change the color.

8014 The color temperature value in kelvins SHALL be related to the *ColorTemperatureMireds* attribute in mireds by the  
 8015 relationship

8016 Color temperature in kelvins = 1,000,000 / *ColorTemperatureMireds*, where *ColorTemperatureMireds* is in the range  
 8017 1 to 65279 mireds inclusive, giving a color temperature range from 1,000,000 kelvins to 15.32 kelvins.

8018 The value *ColorTemperatureMireds* = 0x0000 indicates an undefined value. The value *ColorTemperatureMireds* =  
 8019 0xffff indicates an invalid value.

8020 If this attribute is implemented then the *ColorMode* attribute SHALL also be implemented.

8021 **5.2.2.2.1.9 ColorMode Attribute**

8022 The *ColorMode* attribute indicates which attributes are currently determining the color of the device. If either the  
 8023 *CurrentHue* or *CurrentSaturation* attribute is implemented, this attribute SHALL also be implemented, otherwise it  
 8024 is optional.

8025 The value of the *ColorMode* attribute cannot be written directly - it is set upon reception of any command in section  
 8026 5.2.2.3 to the appropriate mode for that command.

8027 **Table 5-5. Values of the ColorMode Attribute**

Attribute Value	Attributes that Determine the Color
0x00	<i>CurrentHue</i> and <i>CurrentSaturation</i>
0x01	<i>CurrentX</i> and <i>CurrentY</i>
0x02	<i>ColorTemperatureMireds</i>

8028 **5.2.2.2.1.10 Options Attribute<sup>73</sup>**

8029 The *Options* attribute is meant to be changed only during commissioning. The *Options* attribute is a bitmap that  
 8030 determines the default behavior of some cluster commands. Each command that is dependent on the *Options* attribute  
 8031 SHALL first construct a temporary Options bitmap that is in effect during the command processing. The temporary  
 8032 Options bitmap has the same format and meaning as the *Options* attribute, but includes any bits that may be overridden  
 8033 by command fields.

8034 Below is the format and description of the *Options* attribute and temporary Options bitmap and the effect on dependent  
 8035 commands.

<sup>73</sup> CCB 2085

8036

Table 5-6. *Options* Attribute

Bit	Name	Values & Summary
0	ExecuteIfOff	0 – Do not execute command if the On/Off cluster, OnOff attribute is 0x00 (FALSE) 1 – Execute command if the On/Off cluster, OnOff attribute is 0x00 (FALSE)

8037

8038 **ExecuteIfOff:** Command execution SHALL NOT continue beyond the *Options* processing if all of these criteria are  
8039 true:

- 8040 • The On/Off cluster exists on the same endpoint as this cluster.
- 8041 • The *OnOff* attribute of the On/Off cluster, on this endpoint, is 0x00 (FALSE).
- 8042 • The value of the ExecuteIfOff bit is 0.

#### 8043 5.2.2.2.1.11 EnhancedCurrentHue Attribute

8044 The EnhancedCurrentHue attribute represents non-equidistant steps along the CIE 1931 color triangle, and it provides  
8045 16-bits precision.

8046 The upper 8 bits of this attribute SHALL be used as an index in the implementation specific XY lookup table to  
8047 provide the non-equidistance steps (see the ZLL test specification for an example). The lower 8 bits SHALL be used  
8048 to interpolate between these steps in a linear way in order to provide color zoom for the user.

8049 To provide compatibility with standard ZCL, the CurrentHue attribute SHALL contain a hue value in the range 0 to  
8050 254, calculated from the EnhancedCurrentHue attribute.

#### 8051 5.2.2.2.1.12 EnhancedColorMode Attribute

8052 The EnhancedColorMode attribute specifies which attributes are currently determining the color of the device, as  
8053 detailed in Table 5-7.

8054 Table 5-7. Values of the *EnhancedColorMode* Attribute

Attribute Value	Attributes That Determine the Color
0x00	CurrentHue and CurrentSaturation
0x01	<i>CurrentX</i> and <i>CurrentY</i>
0x02	<i>ColorTemperatureMireds</i>
0x03	<i>EnhancedCurrentHue</i> and <i>CurrentSaturation</i>

8055

8056 To provide compatibility with standard ZCL, the original ColorMode attribute SHALL indicate ‘CurrentHue and  
8057 CurrentSaturation’ when the light uses the EnhancedCurrentHue attribute. If the ColorMode attribute is changed, e.g.,  
8058 due to one of the standard color control cluster commands defined in the ZCL, its new value SHALL be copied to the  
8059 EnhancedColorMode attribute.

#### 8060 5.2.2.2.1.13 ColorLoopActive Attribute



8061 The ColorLoopActive attribute specifies the current active status of the color loop. If this attribute has the value 0x00,  
 8062 the color loop SHALL not be active. If this attribute has the value 0x01, the color loop SHALL be active. All other  
 8063 values (0x02 – 0xff) are reserved.

8064 **5.2.2.2.1.14 ColorLoopDirection Attribute**

8065 The ColorLoopDirection attribute specifies the current direction of the color loop. If this attribute has the value 0x00,  
 8066 the EnhancedCurrentHue attribute SHALL be decremented. If this attribute has the value 0x01, the  
 8067 EnhancedCurrentHue attribute SHALL be incremented. All other values (0x02 – 0xff) are reserved.

8068 **5.2.2.2.1.15 ColorLoopTime Attribute**

8069 The ColorLoopTime attribute specifies the number of seconds it SHALL take to perform a full color loop, i.e., to  
 8070 cycle all values of the EnhancedCurrentHue attribute (between 0x0000 and 0xffff).

8071 **5.2.2.2.1.16 ColorLoopStartEnhancedHue Attribute**

8072 The ColorLoopStartEnhancedHue attribute specifies the value of the EnhancedCurrentHue attribute from which the  
 8073 color loop SHALL be started.

8074 **5.2.2.2.1.17 ColorLoopStoredEnhancedHue Attribute**

8075 The ColorLoopStoredEnhancedHue attribute specifies the value of the EnhancedCurrentHue attribute before the color  
 8076 loop was started. Once the color loop is complete, the EnhancedCurrentHue attribute SHALL be restored to this value.

8077 **5.2.2.2.1.18 ColorCapabilities Attribute**

8078 The ColorCapabilities attribute specifies the color capabilities of the device supporting the color control cluster, as  
 8079 illustrated in Table 5-8. If a bit is set to 1, the corresponding attributes and commands SHALL become mandatory. If  
 8080 a bit is set to 0, the corresponding attributes and commands need not be implemented.

8081 **Table 5-8. Bit Values of the ColorCapabilities Attribute**

Value	Description	Related Attributes	Mandatory Commands
0	Hue/saturation supported	<i>CurrentHue</i> <i>CurrentSaturation</i>	<i>Move to hue</i> <i>Move hue</i> <i>Step hue</i> <i>Move to saturation</i> <i>Move saturation</i> <i>Step saturation</i> <i>Move to hue and saturation</i> <i>Stop move step</i>
1	Enhanced hue supported <b>Note:</b> hue/saturation must also be supported.	<i>EnhancedCurrentHue</i>	<i>Enhanced move to hue</i> <i>Enhanced move hue</i> <i>Enhanced step hue</i> <i>Enhanced move to hue and saturation</i> <i>Stop move step</i>
2	Color loop supported <b>Note:</b> enhanced hue must also be supported.	<i>ColorLoopActive</i> <i>ColorLoopDirection</i> <i>ColorLoopTime</i> <i>ColorLoopStartEnhancedHue</i> <i>ColorLoopStoredEnhancedHue</i>	<i>Color loop set</i>

Value	Description	Related Attributes	Mandatory Commands
3	XY attributes supported	<i>CurrentX</i> <i>CurrentY</i>	<i>Move to color</i> <i>Move color</i> <i>Step color</i> <i>Stop move step</i>
4	Color temperature supported	<i>ColorTemperatureMireds</i> <i>ColorTempPhysicalMinMireds</i> <i>ColorTempPhysicalMaxMireds</i>	<i>Move to color temperature</i> <i>Move color temperature</i> <i>Step color temperature</i> <i>Stop move step</i>

8082 **Note:** The support of the *CurrentX* and *CurrentY* attributes is mandatory regardless of color capabilities.

8083 On receipt of a unicast color control cluster command that is not supported or a general command which affects a  
8084 color control cluster attribute that is not supported, the device SHALL respond with a default response command with  
8085 a status indicating an unsupported cluster command or unsupported attribute, respectively.

#### 8086 **5.2.2.2.1.19 ColorTempPhysicalMinMireds Attribute**

8087 The *ColorTempPhysicalMinMireds* attribute indicates the minimum mired value supported by the hardware.  
8088 *ColorTempPhysicalMinMireds* corresponds to the maximum color temperature in kelvins supported by the hardware.  
8089  $ColorTempPhysicalMinMireds \leq ColorTemperatureMireds$ .

#### 8090 **5.2.2.2.1.20 ColorTempPhysicalMaxMireds Attribute**

8091 The *ColorTempPhysicalMaxMireds* attribute indicates the maximum mired value supported by the hardware.  
8092 *ColorTempPhysicalMaxMireds* corresponds to the minimum color temperature in kelvins supported by the hardware.  
8093  $ColorTemperatureMireds \leq ColorTempPhysicalMaxMireds$ .

#### 8094 **5.2.2.2.1.21 CoupleColorTempToLevelMinMireds Attribute<sup>74</sup>**

8095 The *CoupleColorTempToLevelMinMireds* attribute specifies a lower bound on the value of the  
8096 *ColorTemperatureMireds* attribute for the purposes of coupling the *ColorTemperatureMireds* attribute to the  
8097 *CurrentLevel* attribute when the *CoupleColorTempToLevel* bit of the *Options* attribute of the *Level Control* cluster is  
8098 equal to 1. When coupling the *ColorTemperatureMireds* attribute to the *CurrentLevel* attribute, this value SHALL  
8099 correspond to a *CurrentLevel* value of 0xfe (100%).

8100 This attribute SHALL be set such that the following relationship exists:

$$8101 \quad ColorTempPhysicalMinMireds \leq CoupleColorTempToLevelMinMireds \leq ColorTemperatureMireds$$

8102 Note that since this attribute is stored as a micro reciprocal degree (mired) value (i.e. color temperature in kelvins =  
8103  $1,000,000 / CoupleColorTempToLevelMinMireds$ ), the *CoupleColorTempToLevel-MinMireds* attribute corresponds  
8104 to an upper bound on the value of the color temperature in kelvins supported by the device.

#### 8105 **5.2.2.2.1.22 StartUpColorTemperatureMireds Attribute<sup>75</sup>**

8106 The *StartUpColorTemperatureMireds* attribute SHALL define the desired startup color temperature value a lamp  
8107 SHALL use when it is supplied with power and this value SHALL be reflected in the *ColorTemperatureMireds*  
8108 attribute. In addition, the *ColorMode* and *EnhancedColorMode* attributes SHALL be set to 0x02 (*color temperature*).  
8109 The values of the *StartUpColorTemperatureMireds* attribute are listed in the table below.

<sup>74</sup> ZLO 1.0

<sup>75</sup> ZLO 1.0

8110 **Table 5-9. Values of the *StartUpColorTemperatureMireds* attribute**

Value	Action on power up
0x0000 – 0xffef	Set the <i>ColorTemperatureMireds</i> attribute to this value.
0xffff	Set the <i>ColorTemperatureMireds</i> attribute to its previous value.

8111 **5.2.2.2.2 Defined Primaries Information Attribute Set**

8112 The Defined Primaries Information attribute set contains the attributes summarized in Table 5-10.

8113 **Table 5-10. Defined Primaries Information Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0010	NumberOfPrimaries	uint8	0x00 – 0x06	R	-	M
0x0011	Primary1X	uint16	0x0000 – 0xfeff	R	-	M <sup>0</sup>
0x0012	Primary1Y	uint16	0x0000 – 0xfeff	R	-	M <sup>0</sup>
0x0013	Primary1Intensity	uint8	0x00 - 0xff	R	-	M <sup>0</sup>
0x0014	Reserved	-	-	-	-	-
0x0015	Primary2X	uint16	0x0000 – 0xfeff	R	-	M <sup>1</sup>
0x0016	Primary2Y	uint16	0x0000 – 0xfeff	R	-	M <sup>1</sup>
0x0017	Primary2Intensity	uint8	0x0000- 0xff	R	-	M <sup>1</sup>
0x0018	Reserved	-	-	-	-	-
0x0019	Primary3X	uint16	0x0000 – 0xfeff	R	-	M <sup>2</sup>
0x001a	Primary3Y	uint16	0x0000 – 0xfeff	R	-	M <sup>2</sup>
0x001b	Primary3Intensity	uint8	0x00 - 0xff	R	-	M <sup>2</sup>

8114 M<sup>i</sup> = Mandatory if the value of the *NumberOfPrimaries* attribute is greater than *i*, otherwise optional.

8115 **5.2.2.2.2.1 *NumberOfPrimaries* Attribute**

8116 The *NumberOfPrimaries* attribute contains the number of color primaries implemented on this device. A value of 0xff  
 8117 SHALL indicate that the number of primaries is unknown.

8118 Where this attribute is implemented, the attributes below for indicating the “x” and “y” color values of the primaries  
 8119 SHALL also be implemented for each of the primaries from 1 to *NumberOfPrimaries*, without leaving gaps.  
 8120 Implementation of the *Primary1Intensity* attribute and subsequent intensity attributes is optional.

8121 **5.2.2.2.2.2 *Primary1X* Attribute**

8122 The *Primary1X* attribute contains the normalized chromaticity value x for this primary, as defined in the CIE xyY  
 8123 Color Space.

8124 The value of x SHALL be related to the *Primary1X* attribute by the relationship

8125  $x = \text{Primary1X} / 65536$  (*Primary1X* in the range 0 to 65279 inclusive)

### 8126 5.2.2.2.3 Primary1Y Attribute

8127 The *Primary1Y* attribute contains the normalized chromaticity value  $y$  for this primary, as defined in the CIE xyY  
8128 Color Space.

8129 The value of  $y$  SHALL be related to the *Primary1Y* attribute by the relationship

8130  $y = \text{Primary1Y} / 65536$  (*Primary1Y* in the range 0 to 65279 inclusive)

### 8131 5.2.2.2.4 Primary1Intensity Attribute

8132 The *Primary1Intensity* attribute contains a representation of the maximum intensity of this primary as defined in the  
8133 Dimming Light Curve in the Ballast Configuration cluster (see 5.3), normalized such that the primary with the highest  
8134 maximum intensity contains the value 0xfe.

8135 A value of 0xff SHALL indicate that this primary is not available.

### 8136 5.2.2.2.5 Remaining Attributes

8137 The *Primary2X*, *Primary2Y*, *Primary2Intensity*, *Primary3X*, *Primary3Y* and *Primary3Intensity* attributes are used to  
8138 represent the capabilities of the 2<sup>nd</sup> and 3<sup>rd</sup> primaries, where present, in the same way as for the *Primary1X*, *Primary1Y*  
8139 and *Primary1Intensity* attributes.

## 8140 5.2.2.3 Additional Defined Primaries Information Attribute Set

8141 The Additional Defined Primaries Information attribute set contains the attributes summarized in Table 5-11.

8142 Table 5-11. Additional Defined Primaries Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0020	Primary4X	uint16	0x0000 – 0xfeff	R	-	M <sup>3</sup>
0x0021	Primary4Y	uint16	0x0000 – 0xfeff	R	-	M <sup>3</sup>
0x0022	Primary4Intensity	uint8	0x00 – 0xff	R	-	M <sup>3</sup>
0x0023	Reserved	-	-	-	-	-
0x0024	Primary5X	uint16	0x0000 – 0xfeff	R	-	M <sup>4</sup>
0x0025	Primary5Y	uint16	0x0000 – 0xfeff	R	-	M <sup>4</sup>
0x0026	Primary5Intensity	uint8	0x00 – 0xff	R	-	M <sup>4</sup>
0x0027	Reserved	-	-	-	-	-
0x0028	Primary6X	uint16	0x0000 – 0xfeff	R	-	M <sup>5</sup>
0x0029	Primary6Y	uint16	0x0000 – 0xfeff	R	-	M <sup>5</sup>
0x002a	Primary6Intensity	uint8	0x00 – 0xff	R	-	M <sup>5</sup>

8143

8144 M<sup>*i*</sup> = Mandatory if the value of the *NumberOfPrimaries* attribute is greater than *i*, otherwise optional.

8145 **5.2.2.2.3.1 Attributes**

8146 The *Primary4X*, *Primary4Y*, *Primary4Intensity*, *Primary5X*, *Primary5Y*, *Primary5Intensity*, *Primary6X*, *Primary6Y*  
 8147 and *Primary6Intensity* attributes represent the capabilities of the 4<sup>th</sup>, 5<sup>th</sup> and 6<sup>th</sup> primaries, where present, in the same  
 8148 way as the *Primary1X*, *Primary1Y* and *Primary1Intensity* attributes.

8149 **5.2.2.2.4 Defined Color Points Settings Attribute Set**

8150 The Defined Color Points Settings attribute set contains the attributes summarized in Table 5-12.

8151 **Table 5-12. Defined Color Points Settings Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Def</b>	<b>M/O</b>
0x0030	WhitePointX	uint16	0x0000 – 0xfeff	RW	-	O
0x0031	WhitePointY	uint16	0x0000 – 0xfeff	RW	-	O
0x0032	ColorPointRX	uint16	0x0000 – 0xfeff	RW	-	O
0x0033	ColorPointRY	uint16	0x0000 – 0xfeff	RW	-	O
0x0034	ColorPointRIntensity	uint8	0x00 – 0xff	RW	-	O
0x0035	Reserved	-	-	-	-	-
0x0036	ColorPointGX	uint16	0x0000 – 0xfeff	RW	-	O
0x0037	ColorPointGY	uint16	0x0000 – 0xfeff	RW	-	O
0x0038	ColorPointGIntensity	uint8	0x00 – 0xff	RW	-	O
0x0039	Reserved	-	-	-	-	-
0x003a	ColorPointBX	uint16	0x0000 – 0xfeff	RW	-	O
0x003b	ColorPointBY	uint16	0x0000 – 0xfeff	RW	-	O
0x003c	ColorPointBIntensity	uint8	0x00 – 0xff	RW	-	O

8152 **5.2.2.2.4.1 WhitePointX Attribute**

8153 The *WhitePointX* attribute contains the normalized chromaticity value x, as defined in the CIE xyY Color Space, of  
 8154 the current white point of the device.

8155 The value of x SHALL be related to the *WhitePointX* attribute by the relationship

8156 
$$x = \text{WhitePointX} / 65536 \quad (\text{WhitePointX in the range 0 to 65279 inclusive})$$

8157 **5.2.2.2.4.2 WhitePointY Attribute**

8158 The *WhitePointY* attribute contains the normalized chromaticity value y, as defined in the CIE xyY Color Space, of  
 8159 the current white point of the device.

8160 The value of  $y$  SHALL be related to the *WhitePointY* attribute by the relationship

8161  $y = \text{WhitePointY} / 65536$  (*WhitePointY* in the range 0 to 65279 inclusive)

#### 8162 **5.2.2.2.4.3 ColorPointRX Attribute**

8163 The *ColorPointRX* attribute contains the normalized chromaticity value  $x$ , as defined in the CIE xyY Color Space, of  
8164 the red color point of the device.

8165 The value of  $x$  SHALL be related to the *ColorPointRX* attribute by the relationship

8166  $x = \text{ColorPointRX} / 65536$  (*ColorPointRX* in the range 0 to 65279 inclusive)

#### 8167 **5.2.2.2.4.4 ColorPointRY Attribute**

8168 The *ColorPointRY* attribute contains the normalized chromaticity value  $y$ , as defined in the CIE xyY Color Space, of  
8169 the red color point of the device.

8170 The value of  $y$  SHALL be related to the *ColorPointRY* attribute by the relationship

8171  $y = \text{ColorPointRY} / 65536$  (*ColorPointRY* in the range 0 to 65279 inclusive)

#### 8172 **5.2.2.2.4.5 ColorPointRIntensity Attribute**

8173 The *ColorPointRIntensity* attribute contains a representation of the relative intensity of the red color point as defined  
8174 in the Dimming Light Curve in the Ballast Configuration cluster (see 5.3), normalized such that the color point with  
8175 the highest relative intensity contains the value 0xfe.

8176 A value of 0xff SHALL indicate an invalid value.

#### 8177 **5.2.2.2.4.6 Remaining Attributes**

8178 The *ColorPointGX*, *ColorPointGY*, *ColorPointGIntensity*, *ColorPointBX*, *ColorPointBY* and, *ColorPointBIntensity*  
8179 attributes are used to represent the chromaticity values and intensities of the green and blue color points, in the same  
8180 way as for the *ColorPointRX*, *ColorPointRY* and *ColorPointRIntensity* attributes.

8181 If any one of these red, green or blue color point attributes is implemented then they SHALL all be implemented.

### 8182 **5.2.2.3 Commands Received**

8183 The command IDs for the Color Control cluster are listed in Table 5-13.

8184 **Table 5-13. Command IDs for the Color Control Cluster**

Command Identifier	Description	M/O
0x00	Move to Hue	M <sup>0</sup>
0x01	Move Hue	M <sup>0</sup>
0x02	Step Hue	M <sup>0</sup>
0x03	Move to Saturation	M <sup>0</sup>
0x04	Move Saturation	M <sup>0</sup>
0x05	Step Saturation	M <sup>0</sup>
0x06	Move to Hue and Saturation	M <sup>0</sup>
0x07	Move to Color	M <sup>3</sup>
0x08	Move Color	M <sup>3</sup>

Command Identifier	Description	M/O
0x09	Step Color	M <sup>3</sup>
0x0a	Move to Color Temperature	M <sup>4</sup>
0x40	Enhanced Move to Hue	M <sup>1</sup>
0x41	Enhanced Move Hue	M <sup>1</sup>
0x42	Enhanced Step Hue	M <sup>1</sup>
0x43	Enhanced Move to Hue and Saturation	M <sup>1</sup>
0x44	Color Loop Set	M <sup>2</sup>
0x47	Stop Move Step	M <sup>0,1,3,4</sup>
0x4b	Move Color Temperature	M <sup>4</sup>
0x4c	Step Color Temperature	M <sup>4</sup>

8185

8186 M<sup>i</sup> = Mandatory if bit *i* of the *ColorCapabilities* attribute is equal to 1, otherwise optional.

### 8187 5.2.2.3.1 Generic Usage Notes

8188 If one of these commands is received while the device is in its off state, i.e., the OnOff attribute of the on/off cluster  
 8189 is equal to 0x00, the command SHALL be ignored.

8190 When asked to change color via one of these commands, the implementation SHALL select a color, within the limits  
 8191 of the hardware of the device, which is as close as possible to that requested. The determination as to the true  
 8192 representations of color is out of the scope of this specification.

8193 If a color loop is active (i.e., the ColorLoopActive attribute is equal to 0x01), it SHALL only be stopped by sending a  
 8194 specific color loop set command frame with a request to deactivate the color loop (i.e., the color loop SHALL not be  
 8195 stopped on receipt of another command such as the enhanced move to hue command). In addition, while a color loop  
 8196 is active, a manufacturer MAY choose to ignore incoming color commands which affect a change in hue.

8197 For the move hue command, the Rate field specifies the rate of movement in steps per second. A step is a change in  
 8198 the device's hue of one unit. If the move mode field is equal to 0x01 (Up) or 0x03 (Down) and the Rate field has a  
 8199 value of zero, the command has no effect and a default response command (see 2.4.12) is sent in response, with the  
 8200 status code set to INVALID\_FIELD.

8201 For the move to color temperature command, if the target color specified is not achievable by the hardware then the  
 8202 color temperature SHALL be clipped at the physical minimum or maximum achievable, depending on the color  
 8203 temperature transition, when the device reaches that color temperature (which MAY be before the requested transition  
 8204 time), and a ZCL default response command SHALL be generated, where not disabled, with a status code equal to  
 8205 SUCCESS.

### 8206 5.2.2.3.2 Note on Change of ColorMode

8207 The first action taken when any one of these commands is received is to change the *ColorMode* attribute to the  
 8208 appropriate value for the command (see individual commands). Note that, when moving from one color mode to  
 8209 another (e.g., *CurrentX/CurrentY* to *CurrentHue/CurrentSaturation*), the starting color for the command is formed by  
 8210 calculating the values of the new attributes (in this case *CurrentHue*, *CurrentSaturation*) from those of the old  
 8211 attributes (in this case *CurrentX* and *CurrentY*).

8212 When moving from a mode to another mode that has a more restricted color range (e.g., *CurrentX/CurrentY* to  
8213 *CurrentHue/CurrentSaturation*, or *CurrentHue/CurrentSaturation* to *ColorTemperatureMireds*) it is possible for the  
8214 current color value to have no equivalent in the new mode. The behavior in such cases is manufacturer dependent, and  
8215 therefore it is recommended to avoid color mode changes of this kind during usage.

### 8216 **5.2.2.3.3 Use of the OptionsMask & OptionsOverride fields<sup>76</sup>**

8217 The OptionsMask & OptionsOverride fields SHALL both be present or both omitted in the command. A temporary  
8218 Options bitmap SHALL be created from the *Options* attribute, using the OptionsMask & OptionsOverride fields, if  
8219 present. Each bit of the temporary Options bitmap SHALL be determined as follows:

8220 Each bit in the *Options* attribute SHALL determine the corresponding bit in the temporary Options bitmap, unless the  
8221 OptionsMask field is present and has the corresponding bit set to 1, in which case the corresponding bit in the  
8222 OptionsOverride field SHALL determine the corresponding bit in the temporary Options bitmap.

8223 The resulting temporary Options bitmap SHALL then be processed as defined in section 5.2.2.2.1.10.

### 8224 **5.2.2.3.4 Move to Hue Command**

#### 8225 **5.2.2.3.4.1 Payload Format**

8226 The Move to Hue command payload SHALL be formatted as illustrated in Figure 5-2.

8227 **Figure 5-2. Format of the Move to Hue Command Payload**

Octets	1	1	2	0/1	0/1
Data Type	uint8	enum8	uint16	map8	map8
Field Name	Hue	Direction	Transition Time	OptionsMask	OptionsOverride

#### 8228 **5.2.2.3.4.2 Hue Field**

8229 The Hue field specifies the hue to be moved to.

#### 8230 **5.2.2.3.4.3 Direction Field**

8231 The Direction field SHALL be one of the non-reserved values in Table 5-14.

8232 **Table 5-14. Values of the Direction Field**

Fade Mode Value	Description
0x00	Shortest distance
0x01	Longest distance
0x02	Up
0x03	Down

#### 8233 **5.2.2.3.4.4 Transition Time Field**

8234 The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to move to the new hue.

<sup>76</sup> CCB 2085



8235 **5.2.2.3.4.5 OptionsMask and OptionsOverride fields<sup>77</sup>**

8236 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8237 **5.2.2.3.4.6 Effect on Receipt**

8238 On receipt of this command, a device SHALL also set the *ColorMode* attribute to the value 0x00 and then SHALL  
 8239 move from its current hue to the value given in the Hue field.

8240 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new hue SHALL be  
 8241 equal to the Transition Time field.

8242 As hue is effectively measured on a circle, the new hue MAY be moved to in either direction. The direction of hue  
 8243 change is given by the Direction field. If Direction is 'Shortest distance', the direction is taken that involves the shortest  
 8244 path round the circle. This case corresponds to expected normal usage. If Direction is 'Longest distance', the direction  
 8245 is taken that involves the longest path round the circle. This case can be used for 'rainbow effects'. In both cases, if  
 8246 both distances are the same, the Up direction SHALL be taken.

8247 **5.2.2.3.5 Move Hue Command**

8248 **5.2.2.3.5.1 Payload Format**

8249 The Move Hue command payload SHALL be formatted as illustrated in Figure 5-3.

8250 **Figure 5-3. Format of the Move Hue Command Payload**

<b>Octets</b>	1	1	0/1	0/1
<b>Data Type</b>	enum8	uint8	map8	map8
<b>Field Name</b>	Move Mode	Rate	OptionsMask	OptionsOverride

8251 **5.2.2.3.5.2 Move Mode Field**

8252 The Move Mode field SHALL be one of the non-reserved values in Table 5-15.

8253 **Table 5-15. Values of the Move Mode Field**

<b>Fade Mode Value</b>	<b>Description</b>
0x00	Stop
0x01	Up
0x02	Reserved
0x03	Down

8254 **5.2.2.3.5.3 Rate Field**

8255 The Rate field specifies the rate of movement in steps per second. A step is a change in the device's hue of one unit.  
 8256 If the Rate field has a value of zero, the command has no effect and a default response command (see Chapter 2) is  
 8257 sent in response, with the status code set to INVALID\_FIELD.

<sup>77</sup> CCB 2085

8258 **5.2.2.3.5.4 OptionsMask and OptionsOverride field<sup>78</sup>**

8259 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8260 **5.2.2.3.5.5 Effect on Receipt**8261 On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move  
8262 from its current hue in an up or down direction in a continuous fashion, as detailed in Table 5-16.8263 **Table 5-16. Actions on Receipt for Move Hue Command**

Fade Mode	Action on Receipt
Stop	If moving, stop, else ignore the command (i.e., the command is accepted but has no effect). NB This MAY also be used to stop a Move to Hue command, a Move to Saturation command, or a Move to Hue and Saturation command.
Up	Increase the device's hue at the rate given in the Rate field. If the hue reaches the maximum allowed for the device, then proceed to its minimum allowed value.
Down	Decrease the device's hue at the rate given in the Rate field. If the hue reaches the minimum allowed for the device, then proceed to its maximum allowed value.

8264 **5.2.2.3.6 Step Hue Command**8265 **5.2.2.3.6.1 Payload Format**

8266 The Step Hue command payload SHALL be formatted as illustrated in Figure 5-4.

8267 **Figure 5-4. Format of the Step Hue Command Payload**

Octets	1	1	1	0/1	0/1
Data Type	enum8	uint8	uint8	map8	map8
Field Name	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

8268 **5.2.2.3.6.2 Step Mode Field**

8269 The Step Mode field SHALL be one of the non-reserved values in Table 5-17.

8270 **Table 5-17. Values of the Step Mode Field**

Fade Mode Value	Description
0x00	Reserved
0x01	Up
0x02	Reserved

<sup>78</sup> CCB 2085

0x03	Down
------	------

8271 **5.2.2.3.6.3 Step Size Field**

8272 The change to be added to (or subtracted from) the current value of the device’s hue.

8273 **5.2.2.3.6.4 Transition Time Field**

8274 The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the step. A step  
 8275 is a change in the device’s hue of ‘Step size’ units.

8276 **5.2.2.3.6.5 OptionsMask and OptionsOverride fields<sup>79</sup>**

8277 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8278 **5.2.2.3.6.6 Effect on Receipt**

8279 On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move  
 8280 from its current hue in an up or down direction by one step, as detailed in Table 5-18.

8281 **Table 5-18. Actions on Receipt for Step Hue Command**

Fade Mode	Action on Receipt
Up	Increase the device’s hue by one step, in a continuous fashion. If the hue value reaches the maximum value then proceed to the minimum allowed value.
Down	Decrease the device’s hue by one step, in a continuous fashion. If the hue value reaches the minimum value then proceed to the maximum allowed value.

8282 **5.2.2.3.7 Move to Saturation Command**

8283 **5.2.2.3.7.1 Payload Format**

8284 The Move to Saturation command payload SHALL be formatted as illustrated in Figure 5-5.

8285 **Figure 5-5. Format of the Move to Saturation Command Payload**

Octets	1	2	0/1	0/1
Data Type	uint8	uint16	map8	map8
Field Name	Saturation	Transition Time	OptionsMask	OptionsOverride

8286 **5.2.2.3.7.2 OptionsMask and OptionsOverride fields**

8287 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8288 **5.2.2.3.7.3 Effect on Receipt**

<sup>79</sup> CCB 2085

8289 On receipt of this command, a device set the *ColorMode* attribute to the value 0x00 and SHALL then move from its  
8290 current saturation to the value given in the Saturation field.

8291 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new saturation  
8292 SHALL be equal to the Transition Time field, in 1/10ths of a second.

### 8293 5.2.2.3.8 Move Saturation Command

#### 8294 5.2.2.3.8.1 Payload Format

8295 The Move Saturation command payload SHALL be formatted as illustrated in Figure 5-6.

8296 **Figure 5-6. Format of the Move Saturation Command Payload**

<b>Octets</b>	1	1	0/1	0/1
<b>Data Type</b>	enum8	uint8	map8	map8
<b>Field Name</b>	Move Mode	Rate	OptionsMask	OptionsOverride

#### 8297 5.2.2.3.8.2 Move Mode Field

8298 The Move Mode field SHALL be one of the non-reserved values in Table 5-19.

8299 **Table 5-19. Values of the Move Mode Field**

Fade Mode Value	Description
0x00	Stop
0x01	Up
0x02	Reserved
0x03	Down

#### 8300 5.2.2.3.8.3 Rate Field

8301 The Rate field specifies the rate of movement in steps per second. A step is a change in the device's saturation of one  
8302 unit. If the Rate field has a value of zero, the command has no effect and a default response command (see Chapter 2)  
8303 is sent in response, with the status code set to INVALID\_FIELD.

#### 8304 5.2.2.3.8.4 OptionsMask and OptionsOverride fields<sup>80</sup>

8305 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

#### 8306 5.2.2.3.8.5 Effect on Receipt

8307 On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move  
8308 from its current saturation in an up or down direction in a continuous fashion, as detailed in Table 5-20.

<sup>80</sup> CCB 2085

8309

**Table 5-20. Actions on Receipt for Move Saturation Command**

Fade Mode	Action on Receipt
Stop	If moving, stop, else ignore the command (i.e., the command is accepted but has no affect). NB This MAY also be used to stop a Move to Saturation command, a Move to Hue command, or a Move to Hue and Saturation command.
Up	Increase the device’s saturation at the rate given in the Rate field. If the saturation reaches the maximum allowed for the device, stop.
Down	Decrease the device’s saturation at the rate given in the Rate field. If the saturation reaches the minimum allowed for the device, stop.

8310 **5.2.2.3.9 Step Saturation Command**

8311 **5.2.2.3.9.1 Payload Format**

8312 The Step Saturation command payload SHALL be formatted as illustrated in Figure 5-7.

8313 **Figure 5-7. Format of the Step Saturation Command Payload**

Octets	1	1	1	0/1	0/1
Data Type	enum8	uint8	uint8	map8	map8
Field Name	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

8314 **5.2.2.3.9.2 Step Mode Field**

8315 The Step Mode field SHALL be one of the non-reserved values in Table 5-21.

8316 **Table 5-21. Values of the Step Mode Field**

Step Mode Value	Description
0x00	Reserved
0x01	Up
0x02	Reserved
0x03	Down

8317 **5.2.2.3.9.3 Step Size Field**

8318 The change to be added to (or subtracted from) the current value of the device’s saturation.

8319 **5.2.2.3.9.4 Transition Time Field**

8320 The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the step. A step  
 8321 is a change in the device’s saturation of ‘Step size’ units.

8322 **5.2.2.3.9.5 OptionsMask and OptionsOverride fields<sup>81</sup>**

8323 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8324 **5.2.2.3.9.6 Effect on Receipt**8325 On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move  
8326 from its current saturation in an up or down direction by one step, as detailed in Table 5-22.8327 **Table 5-22. Actions on Receipt for Step Saturation Command**

Step Mode	Action on Receipt
Up	Increase the device's saturation by one step, in a continuous fashion. However, if the saturation value is already the maximum value then do nothing.
Down	Decrease the device's saturation by one step, in a continuous fashion. However, if the saturation value is already the minimum value then do nothing.

8328 **5.2.2.3.10 Move to Hue and Saturation Command**8329 **5.2.2.3.10.1 Payload Format**

8330 The Move to Hue and Saturation command payload SHALL be formatted as illustrated in Figure 5-8.

8331 **Figure 5-8. Move to Hue and Saturation Command Payload**

<b>Octets</b>	1	1	2	0/1	0/1
<b>Data Type</b>	uint8	uint8	uint16	map8	map8
<b>Field Name</b>	Hue	Saturation	Transition Time	OptionsMask	OptionsOverride

8332 **5.2.2.3.10.2 OptionsMask and OptionsOverride fields<sup>82</sup>**

8333 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8334 **5.2.2.3.10.3 Effect on Receipt**8335 On receipt of this command, a device SHALL set the *ColorMode* attribute to the value 0x00 and SHALL then move  
8336 from its current hue and saturation to the values given in the Hue and Saturation fields.8337 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new color SHALL  
8338 be equal to the Transition Time field, in 1/10ths of a second.8339 The path through color space taken during the transition is not specified, but it is recommended that the shortest path  
8340 is taken though hue/saturation space, i.e., movement is 'in a straight line' across the hue/saturation disk.<sup>81</sup> CCB 2085<sup>82</sup> CCB 2085

8341 **5.2.2.3.11 Move to Color Command**

8342 **5.2.2.3.11.1 Payload Format**

8343 The Move to Color command payload SHALL be formatted as illustrated in Figure 5-9.

8344 **Figure 5-9. Format of the Move to Color Command Payload**

<b>Octets</b>	2	2	2	0/1	0/1
<b>Data Type</b>	uint16	uint16	uint16	map8	map8
<b>Field Name</b>	ColorX	ColorY	Transition Time	OptionsMask	OptionsOverride

8345 **5.2.2.3.11.2 OptionsMask and OptionsOverride fields**

8346 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8347 **5.2.2.3.11.3 Effect on Receipt**

8348 On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01,  
 8349 and SHALL then move from its current color to the color given in the ColorX and ColorY fields.

8350 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new color SHALL  
 8351 be equal to the Transition Time field, in 1/10ths of a second.

8352 The path through color space taken during the transition is not specified, but it is recommended that the shortest path  
 8353 is taken though color space, i.e., movement is 'in a straight line' across the CIE xyY Color Space.

8354 **5.2.2.3.12 Move Color Command**

8355 **5.2.2.3.12.1 Payload Format**

8356 The Move Color command payload SHALL be formatted as illustrated in Figure 5-10.

8357 **Figure 5-10. Format of the Move Color Command Payload**

<b>Octets</b>	2	2	0/1	0/1
<b>Data Type</b>	int16	int16	map8	map8
<b>Field Name</b>	RateX	RateY	OptionsMask	OptionsOverride

8358 **5.2.2.3.12.2 RateX Field**

8359 The RateX field specifies the rate of movement in steps per second. A step is a change in the device's CurrentX  
 8360 attribute of one unit.

8361 **5.2.2.3.12.3 RateY Field**

8362 The RateY field specifies the rate of movement in steps per second. A step is a change in the device's CurrentY  
 8363 attribute of one unit.

8364 **5.2.2.3.12.4 OptionsMask and OptionsOverride fields<sup>83</sup>**

<sup>83</sup> CCB 2085

8365 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

#### 8366 **5.2.2.3.12.5 Effect on Receipt**

8367 On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01,  
8368 and SHALL then move from its current color in a continuous fashion according to the rates specified. This movement  
8369 SHALL continue until the target color for the next step cannot be implemented on this device.

8370 If both the RateX and RateY fields contain a value of zero, no movement SHALL be carried out, and the command  
8371 execution SHALL have no effect other than stopping the operation of any previously received command of this cluster.  
8372 This command can thus be used to stop the operation of any other command of this cluster.

### 8373 **5.2.2.3.13 Step Color Command**

#### 8374 **5.2.2.3.13.1 Payload Format**

8375 The Step Color command payload SHALL be formatted as illustrated in Figure 5-11.

8376 **Figure 5-11. Format of the Step Color Command Payload**

<b>Octets</b>	2	2	2	0/1	0/1
<b>Data Type</b>	int16	int16	uint16	map8	map8
<b>Field Name</b>	StepX	StepY	Transition Time	OptionsMask	OptionsOverride

#### 8377 **5.2.2.3.13.2 StepX and StepY Fields**

8378 The StepX and StepY fields specify the change to be added to the device's CurrentX attribute and CurrentY attribute  
8379 respectively.

#### 8380 **5.2.2.3.13.3 Transition Time Field**

8381 The Transition Time field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the color change.  
8382 9999

#### 8383 **5.2.2.3.13.4 OptionsMask and OptionsOverride fields<sup>84</sup>**

8384 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

#### 8385 **5.2.2.3.13.5 Effect on Receipt**

8386 On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x01,  
8387 and SHALL then move from its current color by the color step indicated.

8388 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new color SHALL  
8389 be equal to the Transition Time field, in 1/10ths of a second.

8390 The path through color space taken during the transition is not specified, but it is recommended that the shortest path  
8391 is taken though color space, i.e., movement is 'in a straight line' across the CIE xyY Color Space.

8392 Note also that if the required step is larger than can be represented by signed 16-bit integers then more than one step  
8393 command SHOULD be issued.

<sup>84</sup> CCB 2085



8394 **5.2.2.3.14 Move to Color Temperature Command**

8395 **5.2.2.3.14.1 Payload Format**

8396 The Move to Color Temperature command payload SHALL be formatted as illustrated in Figure 5-12.

8397 **Figure 5-12. Move to Color Temperature Command Payload**

<b>Octets</b>	2	2	0/1	0/1
<b>Data Type</b>	uint16	uint16	map8	map8
<b>Field Name</b>	Color Temperature Mireds	Transition Time	OptionsMask	OptionsOverride

8398 **5.2.2.3.14.2 OptionsMask and OptionsOverride fields**

8399 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8400 **5.2.2.3.14.3 Effect on Receipt**

8401 On receipt of this command, a device SHALL set the value of the *ColorMode* attribute, where implemented, to 0x02,  
 8402 and SHALL then move from its current color to the color given by the Color Temperature Mireds field.

8403 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new color SHALL  
 8404 be equal to the Transition Time field, in 1/10ths of a second.

8405 By definition of this color mode, the path through color space taken during the transition is along the ‘Black Body  
 8406 Line’.

8407 **5.2.2.3.15 Enhanced Move to Hue Command**

8408 The Enhanced Move to Hue command allows lamps to be moved in a smooth continuous transition from their current  
 8409 hue to a target hue.

8410 The payload of this command SHALL be formatted as illustrated in Figure 5-13.

8411 **Figure 5-13. Format of the Enhanced Move to Hue Command**

<b>Octets</b>	2	1	2	0/1	0/1
<b>Data Type</b>	uint16	enum8	uint16	map8	map8
<b>Field Name</b>	Enhanced Hue	Direction	Transition Time	OptionsMask	OptionsOverride

8412 **5.2.2.3.15.1 Enhanced Hue Field**

8413 The Enhanced Hue field is 16-bits in length and specifies the target extended hue for the lamp.

8414 **5.2.2.3.15.2 Direction Field**

8415 This field is identical to the Direction field of the Move to Hue command of the Color Control cluster (see sub-clause  
 8416 5.2.2.3.3).

8417 **5.2.2.3.15.3 Transition Time Field**

8418 This field is identical to the Transition Time field of the Move to Hue command of the Color Control cluster (see sub-  
 8419 clause 5.2.2.3.3).

8420 **5.2.2.3.15.4 OptionsMask and OptionsOverride fields<sup>85</sup>**

8421 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8422 **5.2.2.3.15.5 Effect on Receipt**8423 On receipt of this command, a device SHALL set the ColorMode attribute to 0x00 and set the EnhancedColorMode  
8424 attribute to the value 0x03. The device SHALL then move from its current enhanced hue to the value given in the  
8425 Enhanced Hue field.8426 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new enhanced hue  
8427 SHALL be equal to the Transition Time field.8428 **5.2.2.3.16 Enhanced Move Hue Command**8429 The Enhanced Move Hue command allows lamps to be moved in a continuous stepped transition from their current  
8430 hue to a target hue.

8431 The payload of this command SHALL be formatted as illustrated in Figure 5-14.

8432 **Figure 5-14. Format of the Enhanced Move Hue Command**

<b>Octets</b>	1	2	0/1	0/1
<b>Data Type</b>	enum8	uint16	map8	map8
<b>Field Name</b>	Move Mode	Rate	OptionsMask	OptionsOverride

8433 **5.2.2.3.16.1 Move Mode Field**8434 This field is identical to the Move Mode field of the Move Hue command of the Color Control cluster (see sub-clause  
8435 5.2.2.3.5).8436 **5.2.2.3.16.2 Rate field**8437 The Rate field is 16-bits in length and specifies the rate of movement in steps per second. A step is a change in the  
8438 extended hue of a device by one unit. If the Move Mode field is set to 0x01 (up) or 0x03 (down) and the Rate field  
8439 has a value of zero, the command has no effect and a ZCL Default Response command SHALL be sent in response,  
8440 with the status code set to INVALID\_FIELD. If the Move Mode field is set to 0x00 (stop) the rate field SHALL be  
8441 ignored.8442 **5.2.2.3.16.3 OptionsMask and OptionsOverride fields<sup>86</sup>**

8443 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8444 **5.2.2.3.16.4 Effect on receipt**8445 On receipt of this command, a device SHALL set the ColorMode attribute to 0x00 and set the EnhancedColorMode  
8446 attribute to the value 0x03. The device SHALL then move from its current enhanced hue in an up or down direction  
8447 in a continuous fashion, as detailed in Table 5-23.8448 **Table 5-23. Actions on Receipt of the Enhanced Move Hue Command**

<b>Move Mode</b>	<b>Action on Receipt</b>

<sup>85</sup> CCB 2085<sup>86</sup> CCB 2085

Stop	If moving, stop, else ignore the command (i.e., the command is accepted but has no effect). NB This MAY also be used to stop an Enhanced Move to Hue command or an enhanced Move to Hue and Saturation command.
Up	Increase the device’s enhanced hue at the rate given in the Rate field. If the enhanced hue reaches the maximum allowed for the device, proceed to its minimum allowed value.
Down	Decrease the device’s enhanced hue at the rate given in the Rate field. If the hue reaches the minimum allowed for the device, proceed to its maximum allowed value.

8449 **5.2.2.3.17 Enhanced Step Hue Command**

8450 The Enhanced Step Hue command allows lamps to be moved in a stepped transition from their current hue to a target  
 8451 hue, resulting in a linear transition through XY space.

8452 The payload of this command SHALL be formatted as illustrated in Figure 5-15.

8453 **Figure 5-15. Format of the Enhanced Step Hue Command**

<b>Octets</b>	1	2	2	0/1	0/1
<b>Data Type</b>	enum8	uint16	uint16	map8	map8
<b>Field Name</b>	Step Mode	Step Size	Transition Time	OptionsMask	OptionsOverride

8454 **5.2.2.3.17.1 Step Mode Field**

8455 This field is identical to the Step Mode field of the Step Hue command of the Color Control cluster (see sub-clause  
 8456 5.2.2.3.6).

8457 **5.2.2.3.17.2 Step Size Field**

8458 The Step Size field is 16-bits in length and specifies the change to be added to (or subtracted from) the current value  
 8459 of the device’s enhanced hue.

8460 **5.2.2.3.17.3 Transition Time Field**

8461 The Transition Time field is 16-bits in length and specifies, in units of 1/10ths of a second, the time that SHALL be  
 8462 taken to perform the step. A step is a change to the device’s enhanced hue of a magnitude corresponding to the Step  
 8463 Size field.

8464 **5.2.2.3.17.4 OptionsMask and OptionsOverride fields<sup>87</sup>**

8465 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8466 **5.2.2.3.17.5 Effect on Receipt**

8467 On receipt of this command, a device SHALL set the *ColorMode* attribute to 0x00 and the *EnhancedColorMode*  
 8468 attribute to the value 0x03. The device SHALL then move from its current enhanced hue in an up or down direction  
 8469 by one step, as detailed in Table 5-24.

<sup>87</sup> CCB 2085

8470

**Table 5-24. Actions on Receipt for the Enhanced Step Hue Command**

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

8471 **5.2.2.3.18 Enhanced Move to Hue and Saturation Command**8472 The Enhanced Move to Hue and Saturation command allows lamps to be moved in a smooth continuous transition  
8473 from their current hue to a target hue and from their current saturation to a target saturation.

8474 The payload of this command SHALL be formatted as illustrated in Figure 5-16.

8475 **Figure 5-16. Format of the Enhanced Move to Hue and Saturation Command**

<b>Octets</b>	2	1	2	0/1	0/1
<b>Data Type</b>	uint16	uint8	uint16	map8	map8
<b>Field Name</b>	Enhanced Hue	Saturation	Transition Time	OptionsMask	OptionsOverride

8476 **5.2.2.3.18.1 Enhanced Hue Field**

8477 The Enhanced Hue field is 16-bits in length and specifies the target extended hue for the lamp.

8478 **5.2.2.3.18.2 Saturation Field**8479 This field is identical to the Saturation field of the Move to Hue and Saturation command of the Color Control cluster  
8480 (see sub-clause 5.2.2.3.10).8481 **5.2.2.3.18.3 Transition Time Field**8482 This field is identical to the Transition Time field of the Move to Hue command of the Color Control cluster (see sub-  
8483 clause 5.2.2.3.10).8484 **5.2.2.3.18.4 OptionsMask and OptionsOverride fields**

8485 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8486 **5.2.2.3.18.5 Effect on Receipt**8487 On receipt of this command, a device SHALL set the ColorMode attribute to the value 0x00 and set the  
8488 EnhancedColorMode attribute to the value 0x03. The device SHALL then move from its current enhanced hue and  
8489 saturation to the values given in the enhanced hue and saturation fields.8490 The movement SHALL be continuous, i.e., not a step function, and the time taken to move to the new color SHALL  
8491 be equal to the Transition Time field, in 1/10ths of a second.8492 The path through color space taken during the transition is not specified, but it is recommended that the shortest path  
8493 is taken though XY space, i.e., movement is 'in a straight line' across the hue/saturation disk.

8494 **5.2.2.3.19 Color Loop Set Command**

8495 The Color Loop Set command allows a color loop to be activated such that the color lamp cycles through its range of  
 8496 hues.

8497 The payload of this command SHALL be formatted as illustrated in Figure 5-17.

8498 **Figure 5-17. Format of the Color Loop Set Command**

<b>Octets</b>	1	1	1	2	2	0/1	0/1
<b>Data Type</b>	map8	enum8	enum8	uint16	uint16	map8	map8
<b>Field Name</b>	Update Flags	Action	Direction	Time	Start Hue	OptionsMask	OptionsOverride

8499 **5.2.2.3.19.1 Update Flags Field**

8500 The Update Flags field is 8 bits in length and specifies which color loop attributes to update before the color loop is  
 8501 started. This field SHALL be formatted as illustrated in Figure 5-18.

8502 **Figure 5-18. Format of the Update Flags Field of the Color Loop Set Command**

<b>Bits: 0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4-7</b>
Update Action	Update Direction	Update Time	Update Start Hue	Reserved

8503  
 8504 The Update Action sub-field is 1 bit in length and specifies whether the device SHALL adhere to the action field in  
 8505 order to process the command. If this sub-field is set to 1, the device SHALL adhere to the action field. If this sub-  
 8506 field is set to 0, the device SHALL ignore the action field.

8507 The Update Direction sub-field is 1 bit in length and specifies whether the device SHALL update the  
 8508 ColorLoopDirection attribute with the Direction field. If this sub-field is set to 1, the device SHALL update the value  
 8509 of the ColorLoopDirection attribute with the value of the Direction field. If this sub-field is set to 0, the device SHALL  
 8510 ignore the Direction field.

8511 The Update Time sub-field is 1 bit in length and specifies whether the device SHALL update the ColorLoopTime  
 8512 attribute with the Time field. If this sub-field is set to 1, the device SHALL update the value of the ColorLoopTime  
 8513 attribute with the value of the Time field. If this sub-field is set to 0, the device SHALL ignore the Time field.

8514 The Update Start Hue sub-field is 1 bit in length and specifies whether the device SHALL update the  
 8515 ColorLoopStartEnhancedHue attribute with the Start Hue field. If this sub-field is set to 1, the device SHALL update  
 8516 the value of the ColorLoopStartEnhancedHue attribute with the value of the Start Hue field. If this sub-field is set to  
 8517 0, the device SHALL ignore the Start Hue field.

8518 **5.2.2.3.19.2 Action Field**

8519 The Action field is 8 bits in length and specifies the action to take for the color loop if the Update Action sub-field of  
 8520 the Update Flags field is set to 1. This field SHALL be set to one of the non-reserved values listed in Table 5-25.

8521 **Table 5-25. Values of the Action Field of the Color Loop Set Command**

<b>Value</b>	<b>Description</b>
0x00	De-activate the color loop.

Value	Description
0x01	Activate the color loop from the value in the <i>ColorLoopStartEnhancedHue</i> field.
0x02	Activate the color loop from the value of the <i>EnhancedCurrentHue</i> attribute.

### 8522 5.2.2.3.19.3 Direction Field

8523 The Direction field is 8 bits in length and specifies the direction for the color loop if the Update Direction field of the  
8524 Update Flags field is set to 1. This field SHALL be set to one of the non-reserved values listed in Table 5-26.

8525 **Table 5-26. Values of the Direction Field of the Color Loop Set Command**

Direction Field Value	Description
0x00	Decrement the hue in the color loop.
0x01	Increment the hue in the color loop.

### 8526 5.2.2.3.19.4 Time Field

8527 The Time field is 16 bits in length and specifies the number of seconds over which to perform a full color loop if the  
8528 Update Time field of the Update Flags field is set to 1.

### 8529 5.2.2.3.19.5 Start Hue Field

8530 The Start Hue field is 16 bits in length and specifies the starting hue to use for the color loop if the Update Start Hue  
8531 field of the Update Flags field is set to 1.

### 8532 5.2.2.3.19.6 OptionsMask and OptionsOverride fields<sup>88</sup>

8533 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

### 8534 5.2.2.3.19.7 Effect on Receipt

8535 On receipt of this command, the device SHALL first update its color loop attributes according to the value of the  
8536 Update Flags field, as follows. If the Update Direction sub-field is set to 1, the device SHALL set the  
8537 ColorLoopDirection attribute to the value of the Direction field. If the Update Time sub-field is set to 1, the device  
8538 SHALL set the ColorLoopTime attribute to the value of the Time field. If the Update Start Hue sub-field is set to 1,  
8539 the device SHALL set the ColorLoopStartEnhancedHue attribute to the value of the Start Hue field. If the color loop  
8540 is active (and stays active), the device SHALL immediately react on updates of the ColorLoopDirection and  
8541 ColorLoopTime attributes.

8542 If the Update Action sub-field of the Update Flags field is set to 1, the device SHALL adhere to the action specified  
8543 in the Action field, as follows. If the value of the Action field is set to 0x00 and the color loop is active, i.e. the  
8544 ColorLoopActive attribute is set to 0x01, the device SHALL de-activate the color loop, set the ColorLoopActive  
8545 attribute to 0x00 and set the EnhancedCurrentHue attribute to the value of the ColorLoopStoredEnhancedHue  
8546 attribute. If the value of the Action field is set to 0x00 and the color loop is inactive, i.e. the ColorLoopActive attribute  
8547 is set to 0x00, the device SHALL ignore the action update component of the command. If the value of the action field  
8548 is set to 0x01, the device SHALL set the ColorLoopStoredEnhancedHue attribute to the value of the  
8549 EnhancedCurrentHue attribute, set the ColorLoopActive attribute to 0x01 and activate the color loop, starting from  
8550 the value of the ColorLoopStartEnhancedHue attribute. If the value of the Action field is set to 0x02, the device  
8551 SHALL set the ColorLoopStoredEnhancedHue attribute to the value of the EnhancedCurrentHue attribute, set the  
8552 ColorLoopActive attribute to 0x01 and activate the color loop, starting from the value of the EnhancedCurrentHue  
8553 attribute.

<sup>88</sup> CCB 2085

8554 If the color loop is active, the device SHALL cycle over the complete range of values of the EnhancedCurrentHue  
 8555 attribute in the direction of the ColorLoopDirection attribute over the time specified in the ColorLoopTime attribute.  
 8556 The level of increments/decrements is application specific.

8557 **5.2.2.3.20 Stop Move Step Command**

8558 The Stop Move Step command is provided to allow Move to and Step commands to be stopped. (Note this  
 8559 automatically provides symmetry to the Level Control cluster.)

8560 Note: the Stop Move Step command has no effect on an active color loop.

8561 The Stop Move Step command payload SHALL be formatted as illustrated in Figure 5-19.

8562 **Figure 5-19. Format of the Stop Move Step Command Payload**

<b>Octets</b>	0/1	0/1
<b>Data Type</b>	map8	map8
<b>Field Name</b>	OptionsMask	OptionsOverride

8563 **5.2.2.3.20.1 OptionsMask and OptionsOverride fields<sup>89</sup>**

8564 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8565 **5.2.2.3.20.2 Effect on Receipt**

8566 Upon receipt of this command, any Move to, Move or Step command currently in process SHALL be terminated. The  
 8567 values of the CurrentHue, EnhancedCurrentHue and CurrentSaturation attributes SHALL be left at their present value  
 8568 upon receipt of the Stop Move Step command, and the RemainingTime attribute SHALL be set to zero.

8569 **5.2.2.3.21 Move Color Temperature Command**

8570 The Move Color Temperature command allows the color temperature of a lamp to be moved at a specified rate.

8571 The payload of this command SHALL be formatted as illustrated in Figure 5-20.

8572 **Figure 5-20. Format of the Move Color Temperature Command**

<b>Octets</b>	1	2	2	2	0/1	0/1
<b>Data Type</b>	map8	uint16	uint16	uint16	map8	map8
<b>Field Name</b>	Move Mode	Rate	Color Temperature Minimum Mireds	Color Temperature Maximum Mireds	OptionsMask	OptionsOverride

8573 **5.2.2.3.21.1 Move Mode Field**

8574 This field is identical to the Move Mode field of the Move Hue command of the Color Control cluster (see sub-clause  
 8575 5.2.2.3.5).

8576 **5.2.2.3.21.2 Rate Field**

<sup>89</sup> CCB 2085

8577 The Rate field is 16-bits in length and specifies the rate of movement in steps per second. A step is a change in the  
8578 color temperature of a device by one unit. If the Rate field has a value of zero, the command has no effect and a ZCL  
8579 Default Response command SHALL be sent in response, with the status code set to INVALID\_FIELD.

### 8580 5.2.2.3.21.3 Color Temperature Minimum Mireds Field

8581 The Color Temperature Minimum Mireds field is 16-bits in length and specifies a lower bound on the  
8582 *ColorTemperatureMireds* attribute ( $\equiv$  an upper bound on the color temperature in kelvins) for the current move  
8583 operation such that:

8584  $\text{ColorTempPhysicalMinMireds} \leq \text{Color Temperature Minimum Mireds field} \leq \text{ColorTemperatureMireds}$

8585 As such if the move operation takes the *ColorTemperatureMireds* attribute towards the value of the Color Temperature  
8586 Minimum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Minimum  
8587 Mireds field is set to 0x0000, *ColorTempPhysicalMinMireds* SHALL be used as the lower bound for the  
8588 *ColorTemperatureMireds* attribute.

### 8589 5.2.2.3.21.4 Color Temperature Maximum Mireds Field

8590 The Color Temperature Maximum Mireds field is 16-bits in length and specifies an upper bound on the  
8591 *ColorTemperatureMireds* attribute ( $\equiv$  a lower bound on the color temperature in kelvins) for the current move  
8592 operation such that:

8593  $\text{ColorTemperatureMireds} \leq \text{Color Temperature Maximum Mireds field} \leq \text{ColorTempPhysicalMaxMireds}$

8594 As such if the move operation takes the *ColorTemperatureMireds* attribute towards the value of the Color Temperature  
8595 Maximum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature  
8596 Maximum Mireds field is set to 0x0000, *ColorTempPhysicalMaxMireds* SHALL be used as the upper bound for the  
8597 *ColorTemperatureMireds* attribute.

### 8598 5.2.2.3.21.5 OptionsMask and OptionsOverride fields<sup>90</sup>

8599 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

### 8600 5.2.2.3.21.6 Effect on Receipt

8601 On receipt of this command, a device SHALL set both the *ColorMode* and *EnhancedColorMode* attributes to 0x02.  
8602 The device SHALL then move from its current color temperature in an up or down direction in a continuous fashion,  
8603 as detailed in Table 5-27.

8604 **Table 5-27. Actions on Receipt of the Move Color Temperature Command**

Move Mode	Action on Receipt
Stop	If moving, stop the operation, else ignore the command (i.e., the command is accepted but has no effect).
Up	Increase the <i>ColorTemperatureMireds</i> attribute ( $\equiv$ decrease the color temperature in kelvins) at the rate given in the Rate field. If the <i>ColorTemperatureMireds</i> attribute reaches the maximum allowed for the device (via either the Color Temperature Maximum Mireds field or the <i>ColorTempPhysicalMaxMireds</i> attribute), the move operation SHALL be stopped.
Down	Decrease the <i>ColorTemperatureMireds</i> attribute ( $\equiv$ increase the color temperature in kelvins) at the rate given in the Rate field. If the <i>ColorTemperatureMireds</i> attribute reaches the minimum allowed for the device (via either the Color Temperature Minimum Mireds field or the <i>ColorTempPhysicalMinMireds</i> attribute), the move operation SHALL be stopped.

<sup>90</sup> CCB 2085



8605 **5.2.2.3.22 Step Color Temperature Command**

8606 The Step Color Temperature command allows the color temperature of a lamp to be stepped with a specified step size.  
 8607 The payload of this command SHALL be formatted as illustrated in Figure 5-21.

8608 **Figure 5-21. Format of the Step Color Temperature Command**

<b>Octets</b>	1	2	2	2	2	0/1	0/1
<b>Data Type</b>	map8	uint16	uint16	uint16	uint16	map8	map8
<b>Field Name</b>	Step Mode	Step Size	Transition Time	Color Temperature Minimum Mireds	Color Temperature Maximum Mireds	Options Mask	Options Override

8609 **5.2.2.3.22.1 Step Mode Field**

8610 This field is identical to the Step Mode field of the Step Hue command of the Color Control cluster (see sub-clause  
 8611 5.2.2.3.6).

8612 **5.2.2.3.22.2 Step Size Field**

8613 The Step Size field is 16-bits in length and specifies the change to be added to (or subtracted from) the current value  
 8614 of the device’s color temperature.

8615 **5.2.2.3.22.3 Transition Time Field**

8616 The Transition Time field is 16-bits in length and specifies, in units of 1/10ths of a second, the time that SHALL be  
 8617 taken to perform the step. A step is a change to the device’s color temperature of a magnitude corresponding to the  
 8618 Step Size field.

8619 **5.2.2.3.22.4 Color Temperature Minimum Mireds Field**

8620 The Color Temperature Minimum Mireds field is 16-bits in length and specifies a lower bound on the  
 8621 *ColorTemperatureMireds* attribute ( $\equiv$  an upper bound on the color temperature in kelvins) for the current step  
 8622 operation such that:

8623  $ColorTempPhysicalMinMireds \leq Color\ Temperature\ Minimum\ Mireds\ field \leq ColorTemperatureMireds$

8624 As such if the step operation takes the *ColorTemperatureMireds* attribute towards the value of the Color Temperature  
 8625 Minimum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature Minimum  
 8626 Mireds field is set to 0x0000, *ColorTempPhysicalMinMireds* SHALL be used as the lower bound for the  
 8627 *ColorTemperatureMireds* attribute.

8628 **5.2.2.3.22.5 Color Temperature Maximum Mireds Field**

8629 The Color Temperature Maximum Mireds field is 16-bits in length and specifies an upper bound on the  
 8630 *ColorTemperatureMireds* attribute ( $\equiv$  a lower bound on the color temperature in kelvins) for the current step operation  
 8631 such that:

8632  $ColorTemperatureMireds \leq Color\ Temperature\ Maximum\ Mireds\ field \leq ColorTempPhysicalMaxMireds$

8633 As such if the step operation takes the *ColorTemperatureMireds* attribute towards the value of the Color Temperature  
 8634 Maximum Mireds field it SHALL be clipped so that the above invariant is satisfied. If the Color Temperature  
 8635 Maximum Mireds field is set to 0x0000, *ColorTempPhysicalMaxMireds* SHALL be used as the upper bound for the  
 8636 *ColorTemperatureMireds* attribute.

8637 **5.2.2.3.22.6 OptionsMask and OptionsOverride fields<sup>91</sup>**

8638 The OptionsMask and OptionsOverride fields SHALL be processed according to section 5.2.2.3.3.

8639 **5.2.2.3.22.7 Effect on Receipt**

8640 On receipt of this command, a device SHALL set both the ColorMode and EnhancedColorMode attributes to 0x02.  
 8641 The device SHALL then move from its current color temperature in an up or down direction by one step, as detailed  
 8642 in Table 5-28.

8643 **Table 5-28. Actions on Receipt of the Step Color Temperature Command**

Move Mode	Action on Receipt
Up	Increase the <i>ColorTemperatureMireds</i> attribute ( $\equiv$ decrease the color temperature in kelvins) by one step. If the <i>ColorTemperatureMireds</i> attribute reaches the maximum allowed for the device (via either the Color Temperature Maximum Mireds field or the <i>ColorTempPhysicalMaxMireds</i> attribute), the step operation SHALL be stopped.
Down	Decrease the <i>ColorTemperatureMireds</i> attribute ( $\equiv$ increase the color temperature in kelvins) by one step. If the <i>ColorTemperatureMireds</i> attribute reaches the minimum allowed for the device (via either the Color Temperature Minimum Mireds field or the <i>ColorTempPhysicalMinMireds</i> attribute), the step operation SHALL be stopped.

8644 **5.2.2.4 Commands Generated**

8645 The server generates no cluster specific commands

8646 **5.2.2.5 Scene Table Extensions**

8647 If the Scenes server cluster (see 3.7) is implemented, the following extension fields SHALL be added to the Scenes  
 8648 table in the given order, i.e., the attribute listed as 1 is added first:

- 8649 1. *CurrentX*
- 8650 2. *CurrentY*
- 8651 3. *EnhancedCurrentHue*
- 8652 4. *CurrentSaturation*
- 8653 5. *ColorLoopActive*
- 8654 6. *ColorLoopDirection*
- 8655 7. *ColorLoopTime*
- 8656 8. *ColorTemperatureMireds*

8657 Since there is a direct relation between *ColorTemperatureMireds* and XY, color temperature, if supported, is stored  
 8658 as XY in the scenes table.

8659 Attributes in the scene table that are not supported by the device (according to the *ColorCapabilities* attribute) SHALL  
 8660 be present but ignored.

---

<sup>91</sup> CCB 2085

8661 **5.2.2.6 Attribute Reporting**

8662 This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum  
 8663 and maximum reporting interval and reportable change settings described in the Chapter 2, Foundation. The following  
 8664 attributes SHALL be reportable:

8665 *CurrentX*, *CurrentY*

8666 *CurrentHue* (if implemented), *CurrentSaturation* (if implemented), *ColorTemperatureMireds* (if implemented)

8667 **5.2.3 Client**

8668 The client has no specific dependencies, no specific attributes, and receives no cluster specific commands. The client  
 8669 generates the cluster specific commands detailed in 5.2.2.3, as required by the application.

8670 **5.3 Ballast Configuration Cluster**

8671 **5.3.1 Overview**

8672 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 8673 etc.

8674 This cluster is used for configuring a lighting ballast.

8675 **5.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2104 2193 2230 2393 Deprecated some attributes

8676 **5.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BALCFG	Type 1 (client to server)

8677 **5.3.1.3 Cluster Identifiers**

Identifier	Name
0x0301	Ballast Configuration

8678 **5.3.2 Server**

8679 **5.3.2.1 Dependencies**

8680 For the alarm functionality specified by this cluster to be operational, the Alarms server cluster SHALL be  
 8681 implemented on the same endpoint.

8682 **5.3.2.2 Attributes**

8683 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
8684 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
8685 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
8686 are listed in Table 5-29.

8687 **Table 5-29. Ballast Configuration Attribute Sets**

Attribute Set Identifier	Description
0x000	Ballast information
0x001	Ballast settings
0x002	Lamp information
0x003	Lamp settings

8688

8689 **5.3.2.2.1 Ballast Information Attribute Set**

8690 The Ballast Information attribute set contains the attributes summarized in Table 5-30.

8691 **Table 5-30. Attributes of the Ballast Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	PhysicalMinLevel	uint8	0x01 – 0xfe	R	0x01	M <sup>92</sup>
0x0001	PhysicalMaxLevel	uint8	0x01 – 0xfe	R	0xfe	M <sup>93</sup>
0x0002	BallastStatus	map8	0000 00xx	R	0000 0000	O <sup>94</sup>

8692 **5.3.2.2.1.1 PhysicalMinLevel Attribute**

8693 The *PhysicalMinLevel* attribute is 8 bits in length and specifies the minimum light output<sup>95</sup> the ballast can achieve.  
8694 This attribute SHALL be specified in the range 0x01 to 0xfe, and specifies the light output of the ballast according to  
8695 the dimming light curve (see 5.3.4).

8696 **5.3.2.2.1.2 PhysicalMaxLevel Attribute**

8697 The *PhysicalMaxLevel* attribute is 8 bits in length and specifies the maximum light output<sup>96</sup> the ballast can achieve  
8698 according to the dimming light curve (see 5.3.4).

8699 **5.3.2.2.1.3 BallastStatus Attribute**


---

<sup>92</sup> CCB 2193

<sup>93</sup> CCB 2193

<sup>94</sup> CCB 2193

<sup>95</sup> CCB 2104

<sup>96</sup> CCB 2104

8700 The *BallastStatus* attribute is 8 bits in length and specifies the activity status of the ballast functions. The usage of the  
 8701 bits is specified in Table 5-31. Where a function is active, the corresponding bit SHALL be set to 1. Where a function  
 8702 is not active, the corresponding bit SHALL be set to 0.

8703 **Table 5-31. Bit Usage of the *BallastStatus* Attribute**

Bit	Function	Details
0	Ballast Non-operational	0 = The ballast is fully operational 1 = The ballast is not fully operational
1	Lamp Failure <sup>97</sup>	0 = All lamps are operational 1 = One or more lamp is not in its socket or is faulty

8704 **5.3.2.2.2 Ballast Settings Attribute Set**

8705 The Ballast Settings attribute set contains the attributes summarized in Table 5-32.

8706 **Table 5-32. Attributes of the Ballast Settings Attribute Set**

Id	Name	Type	Range	Acc	Default	M/O
0x0010	MinLevel	uint8	0x01 – 0xfe	RW	<i>PhysicalMinLevel</i>	M
0x0011	MaxLevel	uint8	0x01 – 0xfe	RW	<i>PhysicalMaxLevel</i>	M
0x0012	PowerOnLevel	uint8	0x00 – 0xfe	RW	<i>PhysicalMaxLevel</i>	D <sup>98</sup>
0x0013	PowerOnFadeTime	uint16	0x0000 – 0xffff	RW	0x0000	D <sup>99</sup>
0x0014	IntrinsicBallastFactor	uint8	0x00 – 0xfe	RW	-	O
0x0015	BallastFactorAdjustment	uint8	0x64 – MS	RW	0xff	O

8707 **5.3.2.2.2.1 *MinLevel* Attribute**

8708 The *MinLevel* attribute is 8 bits in length and specifies the light output of the ballast according to the dimming light  
 8709 curve (see 5.3.4) when the Level Control Cluster’s CurrentLevel attribute equals to 0x01 (1) (and the On/Off Clusters’s  
 8710 OnOff attribute equals to 0x01)<sup>100</sup>.

8711 The value of this attribute SHALL be both greater than or equal to *PhysicalMinLevel* and less than or equal to  
 8712 *MaxLevel*. If an attempt is made to set this attribute to a level where these conditions are not met, a default response  
 8713 command SHALL be returned with status code set to INVALID\_VALUE, and the level SHALL not be set.

8714 **5.3.2.2.2.2 *MaxLevel* Attribute**

8715 The *MaxLevel* attribute is 8 bits in length and specifies the light output of the ballast according to the dimming light  
 8716 curve (see 5.3.4) when the Level Control Cluster’s CurrentLevel attribute equals to 0xfe (254) (and the On/Off  
 8717 Cluster’s OnOff attribute equals to 0x01)<sup>101</sup>.

<sup>97</sup> CCB 2230 Updated to align with DALI

<sup>98</sup> CCB 2393 Deprecated: see Chapter 2 for more information

<sup>99</sup> CCB 2393 Deprecated: see Chapter 2 for more information

<sup>100</sup> CCB 2104

<sup>101</sup> CCB 2104

8718 The value of this attribute SHALL be both less than or equal to *PhysicalMaxLevel* and greater than or equal to  
8719 *MinLevel*. If an attempt is made to set this attribute to a level where these conditions are not met, a default response  
8720 command SHALL be returned with status code set to *INVALID\_VALUE*, and the level SHALL not be set.

#### 8721 **5.3.2.2.2.3 IntrinsicBallastFactor Attribute**

8722 The *IntrinsicBallastFactor* attribute is 8 bits in length and specifies as a percentage the ballast factor of the ballast/lamp  
8723 combination (see also 5.3), prior to any adjustment.

8724 A value of 0xff indicates an invalid value.

#### 8725 **5.3.2.2.2.4 BallastFactorAdjustment Attribute**

8726 The *BallastFactorAdjustment* attribute is 8 bits in length and specifies the multiplication factor, as a percentage, to be  
8727 applied to the configured light output of the lamps (see also Overview5.3). A typical usage of this mechanism is to  
8728 compensate for reduction in efficiency over the lifetime of a lamp.

8729 The light output is given by

8730  $Actual\ light\ output = configured\ light\ output \times BallastFactorAdjustment / 100\%$

8731 The range for this attribute is manufacturer dependent. If an attempt is made to set this attribute to a level that cannot  
8732 be supported, a default response command SHALL be returned with status code set to *INVALID\_VALUE*, and the  
8733 level SHALL not be set. The value 0xff indicates that ballast factor scaling is not in use.

### 8734 **5.3.2.2.3 Lamp Information Attribute Set**

8735 The lamp information attribute set contains the attributes summarized in Table 5-33.

8736 **Table 5-33. Attributes of the Lamp Information Attribute Set**

Identifier	Name	Type	Range	Access	Default	M/O
0x0020	LampQuantity	uint8	0x00 – 0xfe	R	-	O

#### 8737 **5.3.2.2.3.1 LampQuantity Attribute**

8738 The *LampQuantity* attribute is 8 bits in length and specifies the number of lamps connected to this ballast. (**Note 1:**  
8739 this number does not take into account whether lamps are actually in their sockets or not).

#### 8740 **5.3.2.2.4 Lamp Settings Attribute Set**

8741 The Lamp Settings attribute set contains the attributes summarized in Table 5-34. If *LampQuantity* is greater than one,  
8742 each of these attributes is taken to apply to the lamps as a set. For example, all lamps are taken to be of the same  
8743 *LampType* with the same *LampBurnHours*.

8744 **Table 5-34. Attributes of the Lamp Settings Attribute Set**

Id	Name	Type	Range	Acc	Default	M/O
0x0030	LampType	string	-	RW	empty string	O
0x0031	LampManufacturer	string	-	RW	empty string	O
0x0032	LampRatedHours	uint24	0x000000 – 0xfffffe	RW	0xffffff	O
0x0033	LampBurnHours	uint24	0x000000 – 0xfffffe	RW	0x000000	O

0x0034	LampAlarmMode	map8	0000 000x	RW	0000 0000	O
0x0035	LampBurnHoursTripPoint	uint24	0x000000 – 0xfffffe	RW	0xffffff	O

8745 **5.3.2.2.4.1 LampType Attribute**

8746 The *LampType* attribute is a character string of up to 16 bytes in length. It specifies the type of lamps (including their  
 8747 wattage) connected to the ballast.

8748 **5.3.2.2.4.2 LampManufacturer Attribute**

8749 The *LampManufacturer* attribute is a character string of up to 16 bytes in length. It specifies the name of the  
 8750 manufacturer of the currently connected lamps.

8751 **5.3.2.2.4.3 LampRatedHours Attribute**

8752 The *LampRatedHours* attribute is 24 bits in length and specifies the number of hours of use the lamps are rated for by  
 8753 the manufacturer.

8754 A value of 0xffffff indicates an invalid or unknown time.

8755 **5.3.2.2.4.4 LampBurnHours Attribute**

8756 The *LampBurnHours* attribute is 24 bits in length and specifies the length of time, in hours, the currently connected  
 8757 lamps have been operated, cumulative since the last re-lamping. Burn hours SHALL not be accumulated if the lamps  
 8758 are off.

8759 This attribute SHOULD be reset to zero (e.g., remotely) when the lamp(s) are changed. If partially used lamps are  
 8760 connected, *LampBurnHours* SHOULD be updated to reflect the burn hours of the lamps.

8761 A value of 0xffffff indicates an invalid or unknown time.

8762 **5.3.2.2.4.5 LampAlarmMode Attribute**

8763 The *LampsAlarmMode* attribute is 8 bits in length and specifies which attributes MAY cause an alarm notification to  
 8764 be generated, as listed in Table 5-35. A ‘1’ in each bit position causes its associated attribute to be able to generate an  
 8765 alarm. (**Note:** All alarms are also logged in the alarm table – see Alarms cluster 3.11).

8766 **Table 5-35. Values of the *MainsAlarmMode* Attribute**

Attribute Bit Number	Attribute
0	LampBurnHours

8767 **5.3.2.2.4.6 LampBurnHoursTripPoint Attribute**

8768 The *LampBurnHoursTripPoint* attribute is 24 bits in length and specifies the number of hours the *LampBurnHours*  
 8769 attribute MAY reach before an alarm is generated.

8770 If the Alarms cluster is not present on the same device this attribute is not used and thus MAY be omitted (see 5.3.2.1).

8771 The Alarm Code field included in the generated alarm SHALL be 0x01.

8772 If this attribute takes the value 0xffffff then this alarm SHALL not be generated.

8773 **5.3.2.3 Commands**

8774 No cluster specific commands are received or generated by the server.

8775 **5.3.3 Client**

8776 The client has no attributes. No cluster specific commands are received by the client. No cluster specific commands  
8777 are generated by the client.

8778 **5.3.4 The Dimming Light Curve**

8779 The dimming curve is recommended to be logarithmic, as defined by the following equation:

$$\%Light = 10 \left( \frac{Level-1}{\left(\frac{255}{3}\right)} \right)^{-1}$$

8780

8781 Where: %Light is the percent light output of the ballast and

8782 Level is an 8-bit integer between 1 (0.1% light output) and 254 (100% output) that is adjusted for MinLevel  
8783 and MaxLevel using the following equation<sup>102</sup>:

8784  $Level = (MaxLevel - MinLevel) * CurrentLevel / 253 + (254 * MinLevel - MaxLevel) / 253.$

8785 **Note 1:** Value 255 is not used.

8786 **Note 2:** The light output is determined by this curve together with the *IntrinsicBallastFactor* and  
8787 *BallastFactorAdjustment* Attributes.

8788 The table below gives a couple of examples of the dimming light curve for different values of MinLevel, MaxLevel  
8789 and CurrentLevel<sup>103</sup>.

8790

**Table 5-36. Examples of The Dimming Light Curve**

MinLevel	MaxLevel	CurrentLevel	Level	%Light
1	254	1	1	0.10%
1	254	10	10	0.13%
1	254	100	100	1.49%
1	254	154	254	100%
170	254	1	170	10.1%
170	254	10	173	11.0%
170	254	100	203	24.8%
170	254	254	254	100%
170	230	1	170	10.1%
170	230	10	172	10.7%

<sup>102</sup> CCB 2104<sup>103</sup> CCB 2104



170	230	100	193	19.2%
170	230	254	230	51.9%

8791



8792

## CHAPTER 6 HVAC

8793  
 8794  
 8795  
 8796

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

8797

### 6.1 General Description

8798

#### 6.1.1 Introduction

8799  
 8800

The clusters specified in this document are for use typically in HVAC applications, but MAY be used in any application domain.

8801

#### 6.1.2 Terms

8802  
 8803  
 8804  
 8805

**4-pipes:** In a 4-pipe HVAC fan coil system, heated and chilled water each have their own supply and return pipes, while in a 2 pipe system they share the same supply and return. With a 4-pipes system, heating and cooling can take place at the same time in different locations of a building. With a 2-pipes system, only heating or cooling can take place in the whole building.

8806  
 8807

**Precooling:** Cooling a building in the early (cooler) part of the day, so that the thermal mass of the building decreases cooling needs in the later (hotter) part of the day.

8808

#### 6.1.3 Cluster List

8809  
 8810

This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of clarification. The clusters defined in this document are listed in Table 6-1:

8811

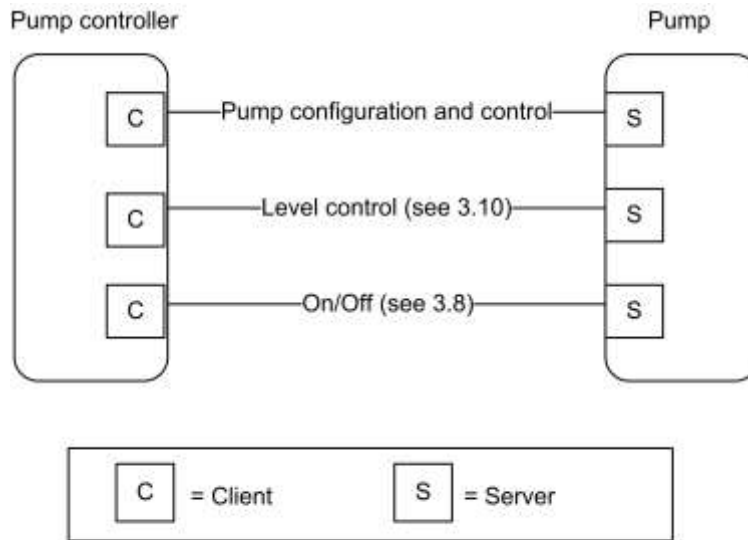
**Table 6-1. Clusters Specified in the HVAC Functional Domain**

ID	Cluster Name	Description
0x0200	Pump Configuration and Control	An interface for configuring and controlling pumps.
0x0201	Thermostat	An interface for configuring and controlling the functionality of a thermostat
0x0202	Fan Control	An interface for controlling a fan in a heating / cooling system
0x0203	Dehumidification Control	An interface for controlling dehumidification
0x0204	Thermostat User Interface Configuration	An interface for configuring the user interface of a thermostat (which MAY be remote from the thermostat)

8812

8813

**Figure 6-1. Typical Usage of Pump Configuration and Control Cluster**



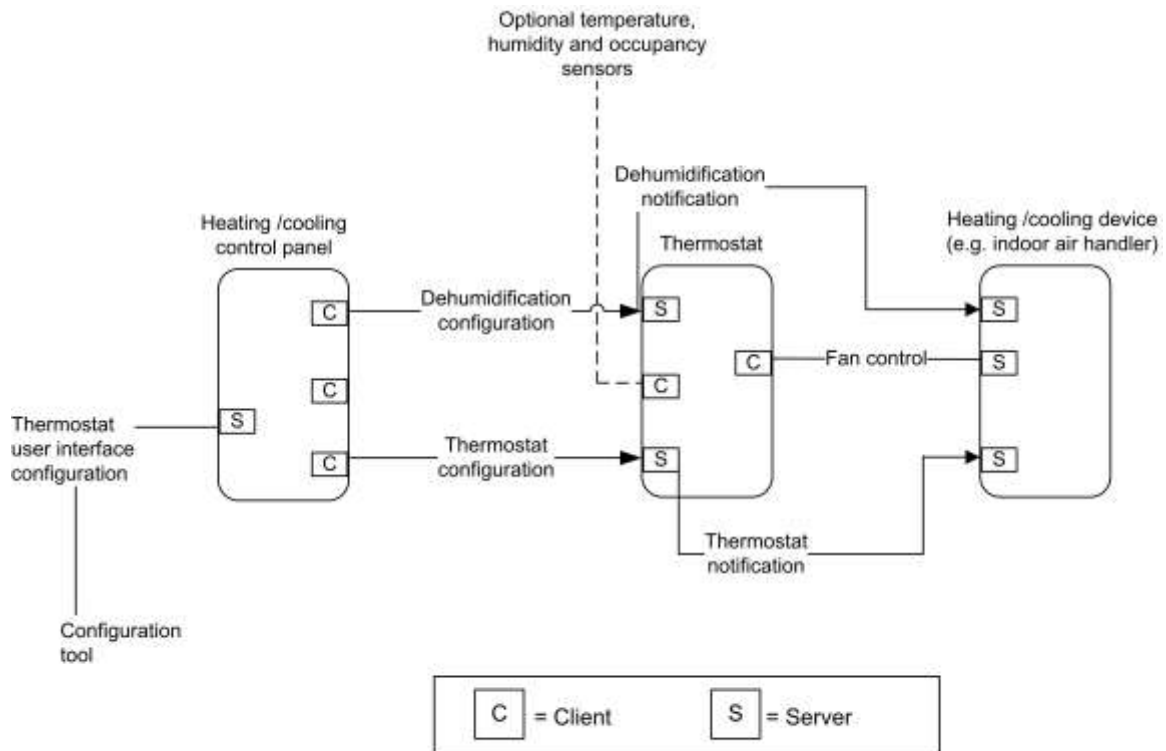
8814

*Note: Device names are examples for illustration purposes only*

8815

8816

**Figure 6-2. Example Usage of the Thermostat and Related Clusters**



8817

*Note: Device names are examples for illustration purposes only*

## 8818 6.2 Pump Configuration and Control

### 8819 6.2.1 Overview

8820 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 8821 etc.

8822 The Pump Configuration and Control cluster provides an interface for the setup and control of pump devices, and the  
 8823 automatic reporting of pump status information. Note that control of pump speed is not included – speed is controlled  
 8824 by the On/Off and Level Control clusters.

#### 8825 6.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

#### 8826 6.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PCC	Type 2 (server to client)

#### 8827 6.2.1.3 Cluster Identifiers

Identifier	Name
0x0200	Pump Configuration and Control

## 8828 6.2.2 Server

### 8829 6.2.2.1 Dependencies

8830 Where external pressure, flow, and temperature measurements are processed by this cluster (see Table 6-8), these are  
 8831 provided by a Pressure Measurement cluster (4.5), a Flow Measurement cluster (4.6), and a Temperature Measurement  
 8832 client cluster (4.4), respectively. These 3 client clusters are used for connection to a remote sensor device. The pump  
 8833 is able to use the sensor measurement provided by a remote sensor for regulation of the pump speed.

8834 For the alarms, described in Table 6-9, to be operational, the Alarms server cluster (3.11) SHALL be implemented on  
 8835 the same endpoint.

8836 Note that control of the pump setpoint is not included in this cluster – the On/Off and Level Control clusters (see  
 8837 Figure 6-1) MAY be used by a pump device to turn it on and off and control its setpoint. Note that the Pump  
 8838 Configuration and Control Cluster MAY override on/off/setpoint settings for specific operation modes (See section  
 8839 6.2.2.3.1 for detailed description of the operation and control of the pump.).

### 8840 6.2.2.2 Attributes

8841 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 8842 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 8843 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 8844 are listed in Table 6-2.

8845

Table 6-2. Pump Configuration Attribute Sets

Attribute Set Identifier	Description
0x000	Pump Information
0x001	Pump Dynamic Information
0x002	Pump Settings

8846 **6.2.2.2.1 Pump Information Attribute Set**

8847 The pump information attribute set contains the attributes summarized in Table 6-3:

8848 Table 6-3. Attributes of the Pump Information Attribute Set

Id	Name	Type	Range	Access	Def	M/O
0x0000	<i>MaxPressure</i>	int16	0x8001-0x7fff	R	-	M
0x0001	<i>MaxSpeed</i>	uint16	0x0000 – 0xfffe	R	-	M
0x0002	<i>MaxFlow</i>	uint16	0x0000 – 0xfffe	R	-	M
0x0003	<i>MinConstPressure</i>	int16	0x8001-0x7fff	R	-	O
0x0004	<i>MaxConstPressure</i>	int16	0x8001-0x7fff	R	-	O
0x0005	<i>MinCompPressure</i>	int16	0x8001-0x7fff	R	-	O
0x0006	<i>MaxCompPressure</i>	int16	0x8001-0x7fff	R	-	O
0x0007	<i>MinConstSpeed</i>	uint16	0x0000 – 0xfffe	R	-	O
0x0008	<i>MaxConstSpeed</i>	uint16	0x0000 – 0xfffe	R	-	O
0x0009	<i>MinConstFlow</i>	uint16	0x0000 – 0xfffe	R	-	O
0x000a	<i>MaxConstFlow</i>	uint16	0x0000 – 0xfffe	R	-	O
0x000b	<i>MinConstTemp</i>	int16	0x954d – 0x7fff	R	-	O
0x000c	<i>MaxConstTemp</i>	int16	0x954d – 0x7fff	R	-	O

8849 **6.2.2.2.1.1 MaxPressure Attribute**8850 The *MaxPressure* attribute specifies the maximum pressure the pump can achieve. It is a physical limit, and does not  
8851 apply to any specific control mode or operation mode.8852 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8853 display the invalid value.8854 Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa).  
8855 The value -3,276.8 kPa (0x8000) indicates that this value is invalid.8856 **6.2.2.2.1.2 MaxSpeed Attribute**8857 The *MaxSpeed* attribute specifies the maximum speed the pump can achieve. It is a physical limit, and does not apply  
8858 to any specific control mode or operation mode.

8859 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8860 display the invalid value.

8861 Valid range is 0 to 65,534 RPM (steps of 1 RPM).  
8862 The value 65,535 RPM (0xffff) indicates that this value is invalid.

#### 8863 **6.2.2.2.1.3 MaxFlow Attribute**

8864 The *MaxFlow* attribute specifies the maximum flow the pump can achieve. It is a physical limit, and does not apply  
8865 to any specific control mode or operation mode.

8866 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8867 display the invalid value.

8868 Valid range is 0 m<sup>3</sup>/h to 6,553.4 m<sup>3</sup>/h (steps of 0.1 m<sup>3</sup>/h).  
8869 The value 6,553.5 m<sup>3</sup>/h (0xffff) indicates that this value is invalid.

#### 8870 **6.2.2.2.1.4 MinConstPressure Attribute**

8871 The *MinConstPressure* attribute specifies the minimum pressure the pump can achieve when it is running and working  
8872 in control mode constant pressure (*ControlMode* attribute of the Pump settings attribute set is set to Constant pressure).

8873 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8874 display the invalid value.

8875 Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa).  
8876 The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

#### 8877 **6.2.2.2.1.5 MaxConstPressure Attribute**

8878 The *MaxConstPressure* attribute specifies the maximum pressure the pump can achieve when it is working in control  
8879 mode constant pressure (*ControlMode* attribute of the Pump settings attribute set is set to Constant pressure).

8880 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8881 display the invalid value.

8882 Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa).  
8883 The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

#### 8884 **6.2.2.2.1.6 MinCompPressure Attribute**

8885 The *MinCompPressure* attribute specifies the minimum compensated pressure the pump can achieve when it is  
8886 running and working in control mode Proportional pressure (*ControlMode* attribute of the Pump settings attribute set  
8887 is set to Proportional pressure).

8888 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8889 display the invalid value.

8890 Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa).  
8891 The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

#### 8892 **6.2.2.2.1.7 MaxCompPressure Attribute**

8893 The *MaxCompPressure* attribute specifies the maximum compensated pressure the pump can achieve when it is  
8894 working in control mode Proportional pressure (*ControlMode* attribute of the Pump settings attribute set is set to  
8895 Proportional pressure).

8896 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8897 display the invalid value.

8898 Valid range is -3,276.7 kPa to 3,276.7 kPa (steps of 0.1 kPa).  
8899 The value -3,276.8 kPa (0x8000) indicates that this value is invalid.

**8900 6.2.2.2.1.8 *MinConstSpeed* Attribute**

8901 The *MinConstSpeed* attribute specifies the minimum speed the pump can achieve when it is running and working in  
8902 control mode Constant speed (*ControlMode* attribute of the Pump settings attribute set is set to Constant speed).

8903 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8904 display the invalid value.

8905 Valid range is 0 to 65,534 RPM (steps of 1 RPM).  
8906 The value 65,535 RPM (0xffff) indicates that this value is invalid.

**8907 6.2.2.2.1.9 *MaxConstSpeed* Attribute**

8908 The *MaxConstSpeed* attribute specifies the maximum speed the pump can achieve when it is working in control mode  
8909 Constant speed (*ControlMode* attribute of the Pump settings attribute set is set to Constant speed).

8910 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8911 display the invalid value.

8912 Valid range is 0 to 65,534 RPM (steps of 1 RPM).  
8913 The value 65,535 RPM (0xffff) indicates that this value is invalid.

**8914 6.2.2.2.1.10 *MinConstFlow* Attribute**

8915 The *MinConstFlow* attribute specifies the minimum flow the pump can achieve when it is running and working in  
8916 control mode Constant flow (*ControlMode* attribute of the Pump settings attribute set is set to Constant flow).

8917 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8918 display the invalid value.

8919 Valid range is 0 m<sup>3</sup>/h to 6,553.4 m<sup>3</sup>/h (steps of 0.1 m<sup>3</sup>/h).  
8920 The value 6,553.5 m<sup>3</sup>/h (0xffff) indicates that this value is invalid.

**8921 6.2.2.2.1.11 *MaxConstFlow* Attribute**

8922 The *MaxConstFlow* attribute specifies the maximum flow the pump can achieve when it is running and working in  
8923 control mode Constant flow (*ControlMode* attribute of the Pump settings attribute set is set to Constant flow).

8924 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8925 display the invalid value.

8926 Valid range is 0 m<sup>3</sup>/h to 6,553.4 m<sup>3</sup>/h (steps of 0.1 m<sup>3</sup>/h).  
8927 The value 6,553.5 m<sup>3</sup>/h (0xffff) indicates that this value is invalid.

**8928 6.2.2.2.1.12 *MinConstTemp* Attribute**

8929 The *MinConstTemp* attribute specifies the minimum temperature the pump can maintain in the system when it is  
8930 running and working in control mode Constant temperature (*ControlMode* attribute of the Pump settings attribute set  
8931 is set to Constant temperature).

8932 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8933 display the invalid value.

8934 Valid range is -273.15 °C to 327.67 °C (steps of 0.01 °C).  
8935 The value -327.68°C (0x8000) indicates that this value is invalid.

**8936 6.2.2.2.1.13 *MaxConstTemp* Attribute**

8937 The *MaxConstTemp* attribute specifies the maximum temperature the pump can maintain in the system when it is  
8938 running and working in control mode Constant temperature (*ControlMode* attribute of the Pump settings attribute set  
8939 is set to Constant temperature).

8940 This attribute is read only, and can only be set by the manufacturer. If the value is not available, this attribute will  
8941 display the invalid value. *MaxConstTemp* SHALL be greater than or equal to *MinConstTemp*



8942 Valid range is  $-273.15\text{ }^{\circ}\text{C}$  to  $327.67\text{ }^{\circ}\text{C}$  (steps of  $0.01\text{ }^{\circ}\text{C}$ ).  
 8943 The value  $-327.68\text{ }^{\circ}\text{C}$  (0x8000) indicates that this value is invalid.

8944 **6.2.2.2.2 Pump Dynamic Information Attribute Set**

8945 The pump dynamic information attribute set contains the attributes summarized in Table 6-4:

8946 **Table 6-4. Attributes of the Pump Dynamic Information Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0010	<i>PumpStatus</i>	map16	-	RP	-	O
0x0011	<i>EffectiveOperationMode</i>	enum8	0x00 – 0xfe	R	-	M
0x0012	<i>EffectiveControlMode</i>	enum8	0x00 – 0xfe	R	-	M
0x0013	<i>Capacity</i>	int16	0x0000-0x7fff	RP	-	M
0x0014	<i>Speed</i>	uint16	0x0000 - 0xffffe	R	-	O
0x0015	<i>LifetimeRunningHours</i>	uint24	0x000000 - 0xffffffe	RW	0	O
0x0016	<i>Power</i>	uint24	0x000000 - 0xffffffe	RW	-	O
0x0017	<i>LifetimeEnergyConsumed</i>	uint32	0x00000000 - 0xfffffffefe	R	0	O

8947 **6.2.2.2.2.1 PumpStatus Attribute**

8948 The *PumpStatus* attribute specifies the activity status of the pump functions listed in Table 6-5. Where a pump  
 8949 controller function is active, the corresponding bit SHALL be set to 1. Where a pump controller function is not active,  
 8950 the corresponding bit SHALL be set to 0.

8951 **Table 6-5. Values of the PumpStatus Attribute**

Bit	Function	Remarks
0	Device fault	A fault related to the pump device is detected (Corresponds to a Alarm code in the range 6-13, see Table 6-9)
1	Supply fault	A fault related to the supply to the pump is detected (Corresponds to a Alarm code in the range 0-5 or 13, see Table 6-9)
2	Speed low	Setpoint is too low to achieve
3	Speed high	Setpoint is too high to achieve
4	Local override	The pump is overridden by local control
5	Running	Pump is currently running
6	Remote Pressure	A remote pressure sensor is used as the sensor for the regulation of the pump. <i>EffectiveControlMode</i> is Constant pressure, and the setpoint for the pump is interpreted as a percentage of the range of the remote sensor ( $[\text{MinMeasuredValue} - \text{MaxMeasuredValue}]$ )

Bit	Function	Remarks
7	Remote Flow	A remote flow sensor is used as the sensor for the regulation of the pump. <i>EffectiveControlModeI</i> is Constant flow, and the setpoint for the pump is interpreted as a percentage of the range of the remote sensor ( $[MinMeasuredValue - MaxMeasuredValue]$ )
8	Remote Temperature	A remote temperature sensor is used as the sensor for the regulation of the pump. <i>EffectiveControlModeI</i> is Constant temperature, and setpoint is interpreted as a percentage of the range of the remote sensor ( $[MinMeasuredValue - MaxMeasuredValue]$ )

#### 8952 **6.2.2.2.2 EffectiveOperationMode Attribute**

8953 The *EffectiveOperationMode* attribute specifies current effective operation mode of the pump. The value of the  
8954 *EffectiveOperationMode* attribute is the same as the *OperationMode* attribute of the Pump settings attribute set, except  
8955 when it is overridden locally. See section 6.2.2.2.3.1 for a detailed description of the operation and control of the  
8956 pump.

8957 This attribute is read only.

8958 Valid range is defined by the operation modes listed in Table 6-1.

#### 8959 **6.2.2.2.3 EffectiveControlMode Attribute**

8960 The *EffectiveControlMode* attribute specifies the current effective control mode of the pump.

8961 The *EffectiveControlMode* attribute contains the control mode that currently applies to the pump. It will have the value  
8962 of the *ControlMode* attribute, unless a remote sensor is used as the sensor for regulation of the pump. In this case,  
8963 *EffectiveControlMode* will display Constant pressure, Constant flow or Constant temperature if the remote sensor is  
8964 a pressure sensor, a flow sensor or a temperature sensor respectively, regardless of the value of the *ControlMode*  
8965 attribute.

8966 See section 6.2.2.2.3.1 for detailed description of the operation and control of the pump. This attribute is read only.

8967 Valid range is defined by the control modes listed in Table 6-8.

#### 8968 **6.2.2.2.4 Capacity Attribute**

8969 The *Capacity* attribute specifies the actual capacity of the pump as a percentage of the effective maximum setpoint  
8970 value. It is updated dynamically as the speed of the pump changes.

8971 This attribute is read only. If the value is not available (the measurement or estimation of the speed is done in the  
8972 pump), this attribute will contain the invalid value.

8973 Valid range is 0 % to 163.835% (0.005 % granularity). Although the *Capacity* attribute is a signed value, values of  
8974 capacity less than zero have no physical meaning.  
8975 The value -163.840 % (0x8000) indicates that this value is invalid.

#### 8976 **6.2.2.2.5 Speed Attribute**

8977 The *Speed* attribute specifies the actual speed of the pump measured in RPM. It is updated dynamically as the speed  
8978 of the pump changes.

8979 This attribute is read only. If the value is not available (the measurement or estimation of the speed is done in the  
8980 pump), this attribute will contain the invalid value.

8981 Valid range is 0 to 65.534 RPM.

8982 The value 65.535 RPM (0xffff) indicates that this value is invalid.

#### 8983 **6.2.2.2.6 LifetimeRunningHours Attribute**

8984 The *LifetimeRunningHours* attribute specifies the accumulated number of hours that the pump has been powered and  
 8985 the motor has been running. It is updated dynamically as it increases. It is preserved over power cycles of the pump.  
 8986 if *LifeTimeRunningHours* rises above maximum value it “rolls over” and starts at 0 (zero).

8987 This attribute is writeable, in order to allow setting to an appropriate value after maintenance. If the value is not  
 8988 available, this attribute will contain the invalid value.

8989 Valid range is 0 to 16,777,214 hrs.  
 8990 The value 16,777,215 (0xfffff) indicates that this value is unknown.

8991 **6.2.2.2.7 Power Attribute**

8992 The *Power* attribute specifies the actual power consumption of the pump in Watts. The value of the *Power* attribute is  
 8993 updated dynamically as the power consumption of the pump changes.

8994 This attribute is read only. If the value is not available (the measurement of power consumption is not done in the  
 8995 pump), this attribute will display the invalid value.

8996 Valid range is 0 to 16,777,214 Watts.  
 8997 The value 16,777,215 (0xfffff) indicates that this value is unknown.

8998 **6.2.2.2.8 LifetimeEnergyConsumed Attribute**

8999 The *LifetimeEnergyConsumed* attribute specifies the accumulated energy consumption of the pump through the entire  
 9000 lifetime of the pump in kWh. The value of the *LifetimeEnergyConsumed* attribute is updated dynamically as the energy  
 9001 consumption of the pump increases. If *LifetimeEnergyConsumed* rises above maximum value it “rolls over” and starts  
 9002 at 0 (zero).

9003 This attribute is writeable, in order to allow setting to an appropriate value after maintenance.

9004 Valid range is 0 kWh to 4,294,967,294 kWh.  
 9005 The value 4,294,967,295 (0xfffffff) indicates that this value is unknown.

9006 **6.2.2.2.3 Pump Settings Attribute Set**

9007 The pump settings attribute set contains the attributes summarized in Table 6-6:

9008 **Table 6-6. Attributes of the Pump Settings Attribute Set**

Identifier	Name	Type	Range	Access	Def	M/O
0x0020	<i>OperationMode</i>	enum8	0x00 – 0xfe	RW	0x00	M
0x0021	<i>ControlMode</i>	enum8	0x00 – 0xfe	RW	0x00	O
0x0022	<i>AlarmMask</i>	map16	-	R	-	O

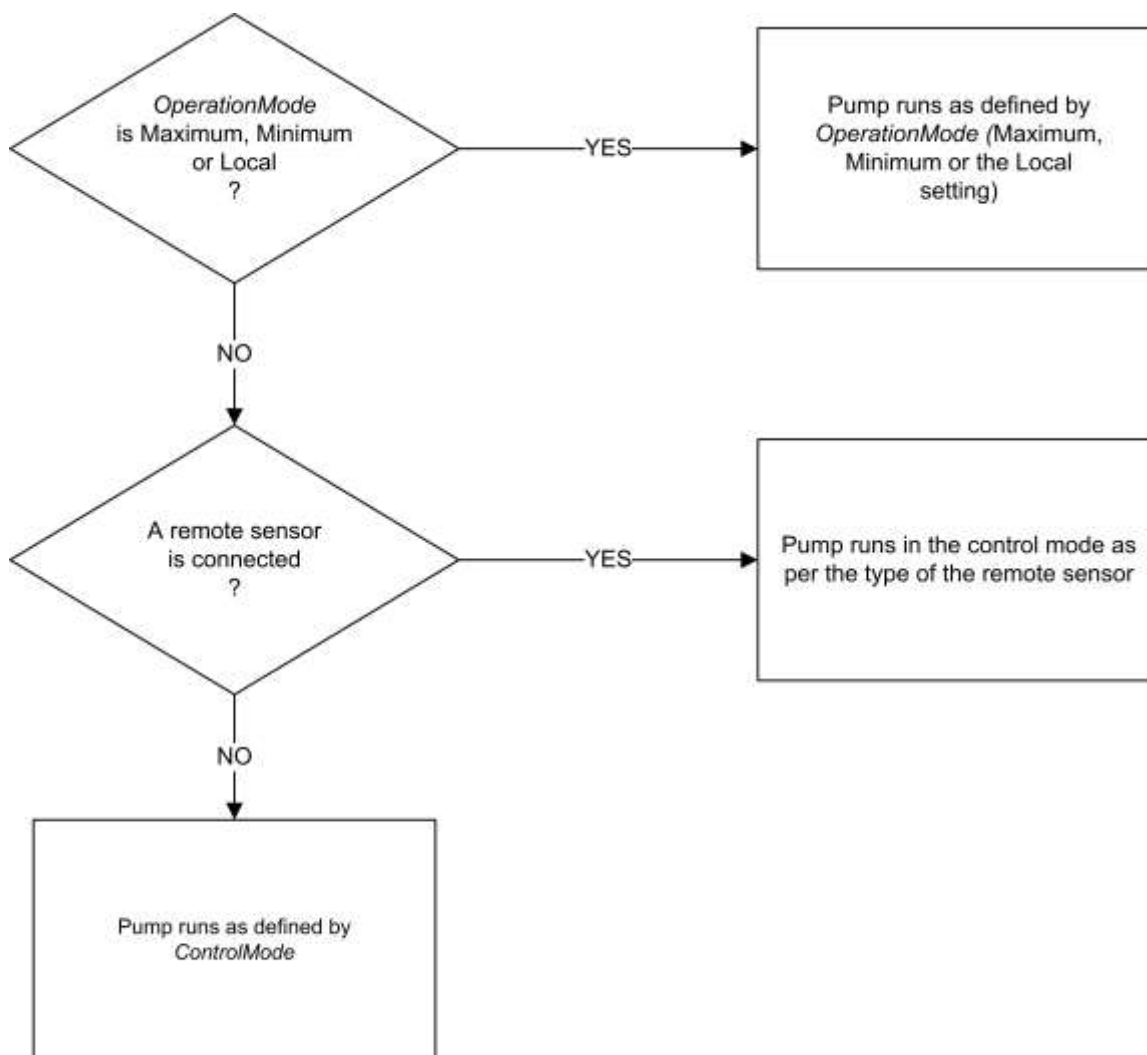
9009 **6.2.2.2.3.1 OperationMode Attribute**

9010 The *OperationMode* attribute specifies the operation mode of the pump. This attribute SHALL have one of the values  
 9011 listed in Table 6-7 Values of the .

9012 The actual operating mode of the pump is a result of the setting of the attributes *OperationMode*, *ControlMode* and  
 9013 the optional connection of a remote sensor. The operation and control is prioritized as shown in the scheme in Figure  
 9014 6-3:

9015

Figure 6-3. Priority Scheme of Pump Operation and Control



9016

9017

9018 If the *OperationMode* attribute is Maximum, Minimum or Local, the *OperationMode* attribute decides how the pump  
9019 is operated.

9020 If the *OperationMode* attribute is Normal and a remote sensor is connected to the pump, the type of the remote sensor  
9021 decides the control mode of the pump. A connected remote pressure sensor will make the pump run in control mode  
9022 Constant pressure and vice versa for flow and temperature type sensors. This is regardless of the setting of the  
9023 *ControlMode* attribute.

9024 If the *OperationMode* attribute is Normal and no remote sensor is connected, the control mode of the pump is decided  
9025 by the *ControlMode* attribute.

9026 *OperationMode* MAY be changed at any time, even when the pump is running. The behavior of the pump at the point  
9027 of changing the value of the *OperationMode* attribute is vendor-specific.

9028

**Table 6-7. Values of the *OperationMode* Attribute**

Value	Name	Explanation
0	Normal	The pump is controlled by a setpoint, as defined by a connected remote sensor or by the <i>ControlMode</i> attribute. (N.B. The setpoint is an internal variable which MAY be controlled between 0% and 100%, e.g., by means of the Level Control cluster 3.10)
1	Minimum	This value sets the pump to run at the minimum possible speed it can without being stopped
2	Maximum	This value sets the pump to run at its maximum possible speed
3	Local	This value sets the pump to run with the local settings of the pump, regardless of what these are

9029 **6.2.2.2.3.2 *ControlMode* Attribute**

9030 The *ControlMode* attribute specifies the control mode of the pump. This attribute SHALL have one of the values listed  
 9031 in Table 6-8 Values of the .

9032 See section 6.2.2.2.3.1 for detailed description of the operation and control of the pump.

9033 *ControlMode* MAY be changed at any time, even when the pump is running. The behavior of the pump at the point  
 9034 of changing is vendor-specific.

9035

**Table 6-8. Values of the *ControlMode* Attribute**

Value	Name	Explanation
0	Constant speed	The pump is running at a constant speed. The setpoint is interpreted as a percentage of the <i>MaxSpeed</i> attribute
1	Constant pressure	The pump will regulate its speed to maintain a constant differential pressure over its flanges. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal pressure sensor, this will be the range derived from the [ <i>MinConstPressure</i> - <i>MaxConstPressure</i> ] attributes. In case of a remote pressure sensor, this will be the range derived from the [ <i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i> ] attributes of the remote pressure sensor.
2	Proportional pressure	The pump will regulate its speed to maintain a constant differential pressure over its flanges. The setpoint is interpreted as a percentage of the range derived of the [ <i>MinCompPressure</i> - <i>MaxCompPressure</i> ] attributes. The internal setpoint will be lowered (compensated) dependant on the flow in the pump (lower flow => lower internal setpoint)
3	Constant flow	The pump will regulate its speed to maintain a constant flow through the pump. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal flow sensor, this will be the range derived from the [ <i>MinConstFlow</i> - <i>MaxConstFlow</i> ] attributes. In case of a remote flow sensor, this will be the range derived from the [ <i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i> ] attributes of the remote flow sensor.
5	Constant temperature	The pump will regulate its speed to maintain a constant temperature. The setpoint is interpreted as a percentage of the range of the sensor used for this control mode. In case of the internal temperature sensor, this will be the range derived from the [ <i>MinConstTemp</i> - <i>MaxConstTemp</i> ] attributes. In case of a remote temperature sensor, this will be the range derived from the [ <i>MinMeasuredValue</i> – <i>MaxMeasuredValue</i> ] attributes of the remote temperature sensor.

Value	Name	Explanation
7	Automatic	The operation of the pump is automatically optimized to provide the most suitable performance with respect to comfort and energy savings. This behavior is manufacturer defined. The pump can be stopped by setting the setpoint of the level control cluster to 0 or by using the On/Off cluster. If the pump is started (at any setpoint), the speed of the pump is entirely determined by the pump.

### 9036 6.2.2.2.3.3 AlarmMask Attribute

9037 The *AlarmMask* attribute specifies whether each of the alarms listed in Table 6-9 is enabled. When the bit number  
9038 corresponding to the alarm code is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding to a code  
9039 in the table (bits 14, 15) are reserved.

9040 When the Alarms cluster is implemented on a device, and one of the alarm conditions included in this table occurs, an  
9041 alarm notification is generated, with the alarm code field set as listed in the table.

9042 **Table 6-9. Alarm Codes**

Alarm Code	Alarm Condition
0	Supply voltage too low
1	Supply voltage too high
2	Power missing phase
3	System pressure too low
4	System pressure too high
5	Dry running
6	Motor temperature too high
7	Pump motor has fatal failure
8	Electronic temperature too high
9	Pump blocked
10	Sensor failure
11	Electronic non fatal failure
12	Electronic fatal failure
13	General fault

### 9043 6.2.2.3 Commands

9044 The server does not receive or generate cluster specific commands.

### 9045 6.2.2.4 Attribute Reporting

9046 This cluster SHALL support attribute reporting using the Report Attributes command, according to the minimum and  
9047 maximum reporting interval, reportable change, and timeout period settings described in the ZCL Foundation  
9048 Specification (see 2.4.7).

9049 The following attributes SHALL be reported:

9050 *PumpStatus*

9051 *Capacity*

## 9052 6.2.3 Client

9053 The client supports no specific attributes. The client does not receive or generate cluster specific commands.

## 9054 6.3 Thermostat

---

### 9055 6.3.1 Overview

9056 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 9057 etc.

9058 This cluster provides an interface to the functionality of a thermostat.

#### 9059 6.3.1.1 Revision History

Rev	Description
1	mandatory global ClusterRevision attribute added; fixed some defaults; CCB 1823, 1480
2	CCB 1981 2186 2249 2250 2251; NFR Thermostat Setback

#### 9060 6.3.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TSTAT	Type 2 (server to client)

#### 9061 6.3.1.3 Cluster Identifiers

Identifier	Name
0x0201	Thermostat

#### 9062 6.3.1.4 Thermostat Temperature Conversion

9063 Many Thermostats store internally or have the capability to display the temperature in degree Fahrenheit format. The  
 9064 Thermostat cluster standardizes temperature representation in degree Celsius format when transferred over the air.  
 9065 Sample code has been provided (see 6.6.2.3). Manufacturers SHOULD use the conversion algorithm provided to  
 9066 convert temperature from Fahrenheit to Celsius and vice versa.

### 9067 6.3.1.5 Thermostat Schedule Feature Mandatory Requirement

9068  
9069 The *StartOfWeek* Attribute is the indicator to show that the Weekly schedule extension is supported. If the Weekly  
9070 schedule extension feature is supported, it is mandatory to also support the *StartOfWeek* Attribute,  
9071 *NumberOfWeeklyTransitions* Attribute, *NumberOfDailyTransitions* Attribute, Set Weekly Schedule Command and  
9072 Get Weekly Schedule Command.

## 9073 6.3.2 Server

### 9074 6.3.2.1 Dependencies

9075 For alarms to be generated by this cluster, the Alarms server cluster (see 3.11) SHALL be included on the same  
9076 endpoint. For remote temperature sensing, the Temperature Measurement client cluster (see 4.4) MAY be included  
9077 on the same endpoint. For occupancy sensing, the Occupancy Sensing client cluster (see 4.8) MAY be included on  
9078 the same endpoint.

### 9079 6.3.2.2 Attributes

9080 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
9081 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
9082 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
9083 for Thermostat are listed in Table 6-10.

9084 **Table 6-10. Currently Defined Thermostat Attribute Sets**

Attribute Set Identifier	Description
0x000	Thermostat Information
0x001	Thermostat Settings
0x002 <sup>104</sup>	Thermostat Schedule & HVAC Relay Attribute Set
0x003	Thermostat Setpoint Change Tracking Attribute Set
0x004	AC Information Attribute Set
0x400 – 0xffff	Reserved for vendor specific attributes

### 9085 6.3.2.2.1 Thermostat Information Attribute Set

9086 The Thermostat Information attribute set contains the attributes summarized in Table 6-11.

9087 **Table 6-11. Attributes of the Thermostat Information Attribute Set**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>LocalTemperature</i>	int16	0x954d – 0x7fff	RP	-	M
0x0001	<i>OutdoorTemperature</i>	int16	0x954d – 0x7fff	R	-	O
0x0002	<i>Occupancy</i>	map8	0000000x	R	00000000	O

<sup>104</sup> CCB 2251 added missing sets 0x002-0x004



Id	Name	Type	Range	Acc	Default	M/O
0x0003	<i>AbsMinHeatSetpointLimit</i>	int16	0x954d – 0x7fff	R	0x02bc (7°C)	O
0x0004	<i>AbsMaxHeatSetpointLimit</i>	int16	0x954d – 0x7fff	R	0x0bb8 (30°C)	O
0x0005	<i>AbsMinCoolSetpointLimit</i>	int16	0x954d – 0x7fff	R	0x0640 (16°C)	O
0x0006	<i>AbsMaxCoolSetpointLimit</i>	int16	0x954d – 0x7fff	R	0x0c80 (32°C)	O
0x0007	<i>PICoolingDemand</i>	uint8	0x00 – 0x64	RP	-	O
0x0008	<i>PIHeatingDemand</i>	uint8	0x00 – 0x64	RP	-	O
0x0009	<i>HVACSystemTypeConfiguration</i>	map8	00xxxxxx	RW	00000000	O

9088 **6.3.2.2.1.1 LocalTemperature Attribute**

9089 *LocalTemperature* represents the temperature in degrees Celsius, as measured locally or remotely (over the network),  
 9090 including any adjustments applied by *LocalTemperatureCalibration* attribute (if any)<sup>105</sup> as follows:

9091  $LocalTemperature = 100 \times (\text{temperature in degrees Celsius} + LocalTemperatureCalibration)$ .

9092 Where  $-273.15^{\circ}\text{C} \leq \text{temperature} \leq 327.67^{\circ}\text{C}$ , corresponding to a *LocalTemperature* in the range 0x954d to 0x7fff.

9093 The maximum resolution this format allows is 0.01 °C.

9094 A *LocalTemperature* of 0x8000 indicates that the temperature measurement is invalid.

9095 **6.3.2.2.1.2 All setpoint attributes in the Thermostat cluster SHALL be**  
 9096 **triggered based off the LocalTemperature attribute (i.e.,**  
 9097 **measured temperature and any calibration offset).**  
 9098 **OutdoorTemperature Attribute**

9099 *OutdoorTemperature* represents the outdoor temperature in degrees Celsius, as measured locally or remotely (over  
 9100 the network). It is measured as described for *LocalTemperature*.

9101 **6.3.2.2.1.3 Occupancy Attribute**

9102 *Occupancy* specifies whether the heated/cooled space is occupied or not, as measured locally or remotely (over the  
 9103 network). If bit 0 = 1, the space is occupied, else it is unoccupied. All other bits are reserved.

9104 **6.3.2.2.1.4 AbsMinHeatSetpointLimit Attribute**

9105 The *MinHeatSetpointLimit* attribute specifies the absolute minimum level that the heating setpoint MAY be set to.  
 9106 This is a limitation imposed by the manufacturer. The value is calculated as described in the *LocalTemperature*  
 9107 attribute.

9108 **6.3.2.2.1.5 AbsMaxHeatSetpointLimit Attribute**

9109 The *MaxHeatSetpointLimit* attribute specifies the absolute maximum level that the heating setpoint MAY be set to.  
 9110 This is a limitation imposed by the manufacturer. The value is calculated as described in the *LocalTemperature*  
 9111 attribute.

9112 **6.3.2.2.1.6 AbsMinCoolSetpointLimit Attribute**

9113 The *MinCoolSetpointLimit* attribute specifies the absolute minimum level that the cooling setpoint MAY be set to.  
 9114 This is a limitation imposed by the manufacturer. The value is calculated as described in the *LocalTemperature*  
 9115 attribute.

<sup>105</sup> NFR Thermostat Setback

9116 **6.3.2.2.1.7 AbsMaxCoolSetpointLimit Attribute**

9117 The *MaxCoolSetpointLimit* attribute specifies the absolute maximum level that the cooling setpoint MAY be set to.  
9118 This is a limitation imposed by the manufacturer. The value is calculated as described in the *LocalTemperature*  
9119 attribute.

9120 **6.3.2.2.1.8 PICoolingDemand Attribute**

9121 The *PICoolingDemand* attribute is 8 bits in length and specifies the level of cooling demanded by the PI (proportional  
9122 integral) control loop in use by the thermostat (if any), in percent. This value is 0 when the thermostat is in “off” or  
9123 “heating” mode.

9124 This attribute is reported regularly and MAY be used to control a heating device.

9125 **6.3.2.2.1.9 PIHeatingDemand Attribute**

9126 The *PIHeatingDemand* attribute is 8 bits in length and specifies the level of heating demanded by the PI loop in  
9127 percent. This value is 0 when the thermostat is in “off” or “cooling” mode.

9128 This attribute is reported regularly and MAY be used to control a cooling device.

9129 **6.3.2.2.1.10 HVACSystemTypeConfiguration Attribute**

9130 The *HVACSystemTypeConfiguration* attribute specifies the HVAC system type controlled by the thermostat. If the  
9131 thermostat uses physical DIP switches to set these parameters, this information SHALL be available read-only from  
9132 the DIP switches. If these parameters are set via software, there SHALL be read/write access in order to provide  
9133 remote programming capability. The meanings of individual bits are detailed in Table 6-12. Each bit represents a type  
9134 of system configuration.

9135 **Table 6-12. HVAC System Type Configuration Values**

Bit Number	Description
0 – 1	<b>Cooling System Stage</b> 00 – Cool Stage 1 01 – Cool Stage 2 10 – Cool Stage 3 11 – Reserved
2 – 3	<b>Heating System Stage</b> 00 – Heat Stage 1 01 – Heat Stage 2 10 – Heat Stage 3 11 – Reserved
4	<b>Heating System Type</b> 0 – Conventional 1 – Heat Pump
5	<b>Heating Fuel Source</b> 0 – Electric / B 1 – Gas / O

9136 **6.3.2.2.2 Thermostat Settings Attribute Set**

9137 The Thermostat settings attribute set contains the attributes summarized in Table 6-13:

9138

**Table 6-13. Attributes of the Thermostat Settings Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>MO</b>
0x0010	<i>LocalTemperatureCalibration</i>	int8	0xE7 – 0x19	RW	0x00 (0°C)	O
0x0011	<i>OccupiedCoolingSetpoint</i>	int16	<i>MinCoolSetpointLimit</i> to <i>MaxCoolSetpointLimit</i>	RWS	0x0a28 (26°C)	M*
0x0012	<i>OccupiedHeatingSetpoint</i>	int16	<i>MinHeatSetpointLimit</i> to <i>MaxHeatSetpointLimit</i>	RWS	0x07d0 (20°C)	M*
0x0013	<i>UnoccupiedCoolingSetpoint</i>	int16	<i>MinCoolSetpointLimit</i> to <i>MaxCoolSetpointLimit</i>	RW	0x0a28 (26°C)	O
0x0014	<i>UnoccupiedHeatingSetpoint</i>	int16	<i>MinHeatSetpointLimit</i> to <i>MaxHeatSetpointLimit</i>	RW	0x07d0 (20°C)	O
0x0015	<i>MinHeatSetpointLimit</i>	int16	0x954d – 0x7fff	RW	0x02bc (7°C)	O
0x0016	<i>MaxHeatSetpointLimit</i>	int16	0x954d – 0x7fff	RW	0x0bb8 (30°C)	O
0x0017	<i>MinCoolSetpointLimit</i>	int16	0x954d – 0x7fff	RW	0x0640 (16°C)	O
0x0018	<i>MaxCoolSetpointLimit</i>	int16	0x954d – 0x7fff	RW	0x0c80 (32°C)	O
0x0019	<i>MinSetpointDeadBand</i>	int8	0x0a – 0x19	RW	0x19 (2.5°C)	O
0x001a	<i>RemoteSensing</i>	map8	00000xxx	RW	0	O
0x001b	<i>ControlSequenceOfOperation</i>	enum8	0x00 – 0x05	RW	0x04	M
0x001c	<i>SystemMode</i>	enum8	See Table 6-16	RWS	0x01	M
0x001d	<i>AlarmMask</i>	map8	00000xxx	R	0	O
0x001e	<i>ThermostatRunningMode</i>	enum8	0x00 – 0x04	R	0x00	O

9139 **\*Note:** "M\*" designates that a server SHALL implement at least one of the attributes designated "M\*." For example,  
 9140 a radiator valve implementing the Thermostat Cluster server would only implement the *OccupiedHeatingSetpoint*  
 9141 attribute. Thermostats SHOULD implement both *OccupiedCoolingSetpoint* and *OccupiedHeatingSetpoint* attributes.  
 9142 The "M\*" designation allows HVAC devices to implement the portions of Thermostat cluster germane to their  
 9143 operation.

9144 **6.3.2.2.2.1 LocalTemperatureCalibration Attribute**

9145 Specifies the offset the Thermostat server SHALL make to the measured temperature (locally or remotely) before  
 9146 calculating, displaying, or communicating the *LocalTemperature* attribute, in steps of 0.1°C.

9147 The purpose of this attribute is to adjust the calibration of the Thermostat server per the user's preferences (e.g., to  
 9148 match if there are multiple servers displaying different values for the same HVAC area) or compensate for variability  
 9149 amongst temperature sensors.

9150 If a Thermostat client attempts to write *LocalTemperatureCalibration* attribute to an unsupported value (e.g., out of  
9151 the range supported by the Thermostat server), the Thermostat server SHALL respond with a Write Attribute Response  
9152 Command with a status of LIMIT\_REACHED and set the value of *LocalTemperatureCalibration* to the upper or  
9153 lower limit reached.<sup>106</sup>

9154

#### 9155 **6.3.2.2.2.2 OccupiedCoolingSetpoint Attribute**

9156 The *OccupiedCoolingSetpoint* attribute specifies the cooling mode setpoint when the room is occupied

9157 The *OccupiedHeatingSetpoint* attribute SHALL always be below the value specified in the *OccupiedCooling*  
9158 *Setpoint* by at least *MinSetpointDeadband*<sup>107</sup>. If an attempt is made to set it such that this condition is violated, a  
9159 response command with the status code INVALID\_VALUE (see 2.5.3) SHALL be returned. If the occupancy status  
9160 of the room is unknown, this attribute SHALL be used as the cooling mode setpoint.

#### 9161 **6.3.2.2.2.3 OccupiedHeatingSetpoint Attribute**

9162 The *OccupiedHeatingSetpoint* attribute specifies the heating mode setpoint when the room is occupied. The  
9163 *OccupiedCoolingSetpoint* attribute SHALL always be above the value specified in the *OccupiedHeatingSetpoint* by  
9164 at least *MinSetpointDeadband*.

9165 If the occupancy status of the room is unknown, this attribute SHALL be used as the heating<sup>108</sup> mode setpoint.

#### 9166 **6.3.2.2.2.4 UnoccupiedCoolingSetpoint Attribute**

9167 The *UnoccupiedCoolingSetpoint* attribute and specifies the cooling mode setpoint when the room is unoccupied. The  
9168 *UnoccupiedHeatingSetpoint* attribute SHALL always be below the value specified in the *UnoccupiedCoolingSetpoint*  
9169 by at least *MinSetpointDeadband*.

9170 If the occupancy status of the room is unknown, this attribute SHALL not be used.

#### 9171 **6.3.2.2.2.5 UnoccupiedHeatingSetpoint Attribute**

9172 The *UnoccupiedHeatingSetpoint* attribute specifies the heating mode setpoint when the room is unoccupiedThe  
9173 *UnoccupiedCoolingSetpoint* attribute SHALL always be above the value specified in the *UnoccupiedHeatingSetpoint*  
9174 by at least *MinSetpointDeadband*.

9175 If the occupancy status of the room is unknown, this attribute SHALL not be used.

#### 9176 **6.3.2.2.2.6 MinHeatSetpointLimit Attribute**

9177 The *MinHeatSetpointLimit* attribute specifies the minimum level that the heating setpoint MAY be set to. If this  
9178 attribute is not present, it SHALL be taken as equal to *AbsMinHeatSetpointLimit*.

9179 This attribute, and the following three attributes, allow the user to define setpoint limits more constrictive than the  
9180 manufacturer imposed ones. Limiting users (e.g., in a commercial building) to such setpoint limits can help conserve  
9181 power.

#### 9182 **6.3.2.2.2.7 MaxHeatSetpointLimit Attribute**

9183 The *MaxHeatSetpointLimit* attribute specifies the maximum level that the heating setpoint MAY be set to. It must be  
9184 less than or equal to *AbsMaxHeatSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to  
9185 *AbsMaxHeatSetpointLimit*.

#### 9186 **6.3.2.2.2.8 MinCoolSetpointLimit Attribute**

---

<sup>106</sup> CCB 1981 & NFR Thermostat Setback

<sup>107</sup> CCB 2251 misnamed *SetpointDeadBand* ( and other places as well)

<sup>108</sup> CCB 2186 typo: should be used for heating (not cooling)

9187 The *MinCoolSetpointLimit* attribute specifies the minimum level that the cooling setpoint MAY be set to. It must be  
 9188 greater than or equal to *AbsMinCoolSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to  
 9189 *AbsMinCoolSetpointLimit*.

9190 **6.3.2.2.2.9 MaxCoolSetpointLimit Attribute**

9191 The *MaxCoolSetpointLimit* attribute specifies the maximum level that the cooling setpoint MAY be set to. . It must  
 9192 be less than or equal to *AbsMaxCoolSetpointLimit*. If this attribute is not present, it SHALL be taken as equal to  
 9193 *AbsMaxCoolSetpointLimit*.

9194 **6.3.2.2.2.10 MinSetpointDeadBand Attribute**

9195 The *MinSetpointDeadBand* attribute specifies the minimum difference between the Heat Setpoint and the Cool  
 9196 SetPoint, in steps of 0.1°C. Its range is 0x0a to 0x19 (1°C to 2.5°C).

9197 **6.3.2.2.2.11 RemoteSensing Attribute**

9198 The RemoteSensing attribute is an 8-bit bitmap that specifies whether the local temperature, outdoor temperature and  
 9199 occupancy are being sensed by internal sensors or remote networked sensors. The meanings of individual bits are  
 9200 detailed in Table 6-14.

9201 **Table 6-14. RemoteSensing Attribute Bit Values**

Bit Number	Description
0	0 – local temperature sensed internally 1 – local temperature sensed remotely
1	0 – outdoor temperature sensed internally 1 – outdoor temperature sensed remotely
2	0 – occupancy sensed internally 1 – occupancy sensed remotely

9202 **6.3.2.2.2.12 ControlSequenceOfOperation Attribute**

9203 The *ControlSequenceOfOperation* attribute specifies the overall operating environment of the thermostat, and thus  
 9204 the possible system modes that the thermostat can operate in. It SHALL be set to one of the non-reserved values in  
 9205 Table 6-15. (**Note:** it is not mandatory to support all values).

9206 **Table 6-15. ControlSequenceOfOperation Attribute Values**

Value	Description	Possible Values of <i>SystemMode</i>
0x00	Cooling Only	Heat and Emergency are not possible
0x01	Cooling With Reheat	Heat and Emergency are not possible
0x02	Heating Only	Cool and precooling (see 6.1.2) are not possible
0x03	Heating With Reheat	Cool and precooling are not possible
0x04	Cooling and Heating 4-pipes (see 1.3.2)	All modes are possible
0x05	Cooling and Heating 4-pipes with Reheat	All modes are possible

9207 **6.3.2.2.2.13 SystemMode Attribute**

9208 The *SystemMode* attribute specifies the current operating mode of the thermostat, It SHALL be set to one of the non-  
 9209 reserved values in Table 6-16, as limited by Table 6-17. (**Note:** It is not mandatory to support all values).

9210

**Table 6-16. SystemMode Attribute Values**

Attribute Value	Description
0x00	Off
0x01	Auto
0x03	Cool
0x04	Heat
0x05	Emergency heating
0x06	Precooling (see 6.1.2)
0x07	Fan only
0x08	Dry
0x09	Sleep

9211 The interpretation of the Heat, Cool and Auto values of *SystemMode* is shown in Table 6-17.

9212

**Table 6-17. Interpretation of SystemMode Values**

Attribute Values	Temperature Below Heat Setpoint	Temperature Between Heat Setpoint and Cool Setpoint	Temperature Above Cool Setpoint
Heat	Temperature below target	Temperature on target	Temperature on target
Cool	Temperature on target	Temperature on target	Temperature above target
Auto	Temperature below target	Temperature on target	Temperature above target

9213 **6.3.2.2.2.14 AlarmMask Attribute**9214 The *AlarmMask* attribute specifies whether each of the alarms listed in Table 6-18 Alarm Codes is enabled. When the  
9215 bit number corresponding to the alarm code is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding  
9216 to a code in the table are reserved.9217 When the Alarms cluster is implemented on a device, and one of the alarm conditions included in this table occurs, an  
9218 alarm notification is generated, with the alarm code field set as listed in the table.

9219

**Table 6-18. Alarm Codes**

Alarm Code	Alarm Condition
0	Initialization failure. The device failed to complete initialization at power-up.
1	Hardware failure
2	Self-calibration failure

9220 **6.3.2.2.2.15 Thermostat Running Mode Attribute**9221 *ThermostatRunningMode* represents the running mode of the thermostat. The thermostat running mode  
9222 can only be Off, Cool or Heat. This attribute is intended to provide additional information when the

9223 thermostat's system mode is in auto mode. The attribute value is maintained to have the same value as  
 9224 the *SystemMode* attribute.

9225 **Table 6.19 Thermostat Running Mode Attribute Values**

Value	Description
0x00	Off
0x03	Cool
0x04	Heat

9226 **6.3.2.2.3 Thermostat Schedule & HVAC Relay Attribute Set**

9227 **Table 6-20. Thermostat Schedule & HVAC Relay Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
Schedule Attribute Set 0x0020 – 0x0028						
x0020	<i>StartOfWeek</i>	enum8	0x00 – 0x06	R	–	O
x0021	<i>NumberOfWeeklyTransitions</i>	uint8	0x00 – 0xff	R	N/A	O
x0022	<i>NumberOfDailyTransitions</i>	uint8	0x00 – 0xff	R	N/A	O
x0023	<i>TemperatureSetpointHold</i>	enum8	0x00 – 0x01	RW	0x00	O
x0024	<i>TemperatureSetpointHoldDuration</i>	uint16	0xffff - 0x05a0	RW	0xffff	O
x0025	<i>ThermostatProgrammingOperationMode</i>	map8	00xxxxxx	RWP <sup>109</sup>	00000000	O
HVAC Relay Attribute Set 0x0029 – 0x002F						
x0029	<i>ThermostatRunningState</i>	map16		R	-	O

9228 **6.3.2.2.3.1 StartOfWeek Attribute**

9229 *StartOfWeek* represents the day of the week that this thermostat considers to be the start of week for weekly set point  
 9230 scheduling. The possible values are given in Table 6-21:

9231 **Table 6-21. StartofWeek Enumeration Values**

Enumeration Field	Value Description
0x00	Sunday
0x01	Monday
0x02	Tuesday
0x03	Wednesday
0x04	Thursday

<sup>109</sup> CCB 2250 attribute specification text requires that this be reportable

Enumeration Field	Value Description
0x05	Friday
0x06	Saturday

9232 If the Weekly schedule extension is supported this attribute SHALL be supported.

9233 This attribute MAY be able to be used as the base to determine if the device supports weekly scheduling by reading  
9234 the attribute. Successful response means that the weekly scheduling is supported.

#### 9235 **6.3.2.2.3.2      NumberOfWeeklyTransitions Attribute**

9236 *NumberOfWeeklyTransitions* attribute determines how many weekly schedule transitions the thermostat is capable of  
9237 handling.

#### 9238 **6.3.2.2.3.3      NumberOfDailyTransitions Attribute**

9239 *NumberOfDailyTransitions* attribute determines how many daily schedule transitions the thermostat is capable of  
9240 handling.

#### 9241 **6.3.2.2.3.4      TemperatureSetpointHold Attribute**

9242 *TemperatureSetpointHold* specifies the temperature hold status on the thermostat, as shown in Table 6-22. If hold  
9243 status is on, the thermostat SHOULD maintain the temperature set point for the current mode until a system mode  
9244 change. If hold status is off, the thermostat SHOULD follow the setpoint transitions specified by its internal scheduling  
9245 program. If the thermostat supports setpoint hold for a specific duration, it SHOULD also implement the  
9246 *TemperatureSetpointHoldDuration* attribute.

9247 **Table 6-22. *TemperatureSetpointHold* Attribute Values**

Enumeration Field	Value Description
0x00	Setpoint Hold Off
0x01	Setpoint Hold On

#### 9248 **6.3.2.2.3.5      TemperatureSetpointHoldDuration Attribute**

9249 *TemperatureSetpointHoldDuration* sets the period in minutes for which a setpoint hold is active. Thermostats that  
9250 support hold for a specified duration SHOULD implement this attribute. The valid range is from 0x0000 – 0x05A0  
9251 (1440 minutes within a day). A value of 0xFFFF indicates the field is unused. All other values are reserved.

#### 9252 **6.3.2.2.3.6      ThermostatProgrammingOperationMode Attribute**

9253 The *ThermostatProgrammingOperationMode* attribute determines the operational state of the thermostat's  
9254 programming. The thermostat SHALL modify its programming operation when this attribute is modified by a client  
9255 and update this attribute when its programming operation is modified locally by a user. The thermostat MAY support  
9256 more than one active *ThermostatProgrammingOperationMode*. For example, the thermostat MAY operate  
9257 simultaneously in Schedule Programming Mode and Recovery Mode. If a thermostat supports Thermostat  
9258 Programming Operation Mode attribute, it SHALL support attribute reporting for this attribute. Any locally-initiated  
9259 changes to the *ThermostatProgrammingOperationMode* SHALL be updated and reported to all clients configured to  
9260 receive such reports. The meanings of individual bits are detailed in Table 6-23. Each bit represents a type of operation.



9261

**Table 6-23. ThermostatProgrammingOperationMode Attribute Values**

Value	Description
0	0 – Simple/setpoint mode. This mode means the thermostat setpoint is altered only by manual up/down changes at the thermostat or remotely, not by internal schedule programming. 1 – Schedule programming mode. This enables or disables any programmed weekly schedule configurations. <i>Note: It does not clear or delete previous weekly schedule programming configurations.</i>
1	0 - Auto/recovery mode set to OFF 1 – Auto/recovery mode set to ON
2	0 – Economy/EnergyStar mode set to OFF 1 – Economy/EnergyStar mode set to ON

9262 **6.3.2.2.3.7 ThermostatRunningState Attribute**

9263 *ThermostatRunningState* represents the current relay state of the heat, cool, and fan relays, whose values are shown in  
 9264 Table 6-24.

9265 **Table 6-24. HVAC Relay State Values**

Bit Number	Description
0	Heat State On
1	Cool State On
2	Fan State On
3	Heat 2 <sup>nd</sup> Stage State On
4	Cool 2 <sup>nd</sup> Stage State On
5	Fan 2 <sup>nd</sup> Stage State On
6	Fan 3 <sup>rd</sup> Stage State On

9266 **6.3.2.2.4 Thermostat Setpoint ChangeTracking Attribute Set**

9267 **Table 6-25. Thermostat Setpoint Change Tracking Attribute Set**

Id	Name	Type	Range	Acc	Def	MO
0x0030	<i>SetpointChangeSource</i>	enum8	0x00 – 0xff	R	0x00	O
0x0031	<i>SetpointChangeAmount</i>	int16	0x0000 – 0xffff	R	0x8000	O
0x0032	<i>SetpointChangeSourceTimestamp</i>	UTC <sup>110</sup>	0x00000000 – 0xffffffe	R	0x00000000	O
0x0034	<i>OccupiedSetback</i> <sup>111</sup>	uint8	<i>OccupiedSetbackMin</i> – <i>OccupiedSetbackMax</i>	RW	0xff	O

<sup>110</sup> CCB 2249 was listed in table as int32 (UTC), though specified as UTC in text

<sup>111111</sup> NFR Thermostat Setback attributes 0x0035-0x003a

Id	Name	Type	Range	Acc	Def	MO
0x0035	<i>OccupiedSetbackMin</i>	uint8	0x00 – <i>OccupiedSetbackMax</i>	R	0xff	O
0x0036	<i>OccupiedSetbackMax</i>	uint8	<i>OccupiedSetbackMin</i> – 0xff	R	0xff	O
0x0037	<i>UnoccupiedSetback</i>	uint8	<i>UnoccupiedSetbackMin</i> – <i>OccupiedSetbackMax</i>	RW	0xff	O
0x0038	<i>UnoccupiedSetbackMin</i>	uint8	0x00 – <i>OccupiedSetbackMax</i>	R	0xff	O
0x0039	<i>UnoccupiedSetbackMax</i>	uint8	<i>OccupiedSetbackMin</i> – 0xff	R	0xff	O
0x003a	<i>EmergencyHeatDelta</i>	uint8	0x00 – 0xff	RW	0xff	O

9268

9269 **6.3.2.2.4.1 SetpointChangeSource Attribute**

9270 The *SetpointChangeSource* attribute specifies the source of the current active *OccupiedCoolingSetpoint* or  
 9271 *OccupiedHeatingSetpoint* (i.e., who or what determined the current setpoint).

9272 *SetpointChangeSource* attribute enables service providers to determine whether changes to setpoints were initiated  
 9273 due to occupant comfort, scheduled programming or some other source (e.g., electric utility or other service provider).  
 9274 Because automation services MAY initiate frequent setpoint changes, this attribute clearly differentiates the source of  
 9275 setpoint changes made at the thermostat.

9276

Table 6-26. *SetpointChangeSource* Values

<i>SetpointChangeSource</i> Attribute	Description
0x00	Manual, user-initiated setpoint change via the thermostat
0x01	Schedule/internal programming-initiated setpoint change
0x02	Externally-initiated setpoint change (e.g., DRLC cluster command, attribute write)

9277 **6.3.2.2.4.2 SetpointChangeAmount Attribute**

9278 The *SetpointChangeAmount* attribute specifies the delta between the current active *OccupiedCoolingSetpoint* or  
 9279 *OccupiedHeatingSetpoint* and the previous active setpoint. This attribute is meant to accompany the  
 9280 *SetpointChangeSource* attribute; devices implementing *SetpointChangeAmount* SHOULD also implement  
 9281 *SetpointChangeSource*.

9282

Table 6-27. *SetpointChangeAmount* Values

<i>SetpointChangeAmount</i> Attribute	Description
0x0000 – 0xffff	The signed difference in 0.01 degrees Celsius between the previous temperature setpoint and the new temperature setpoint.

9283 **6.3.2.2.4.3 SetpointChangeSourceTimestamp Attribute**

9284 The *SetpointChangeSourceTimestamp* attribute specifies the time in UTC at which the *SetpointChangeSourceAmount*  
 9285 attribute change was recorded.

9286 **6.3.2.2.4.4 OccupiedSetback Attribute**

9287 Specifies the degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *LocalTemperature*  
9288 attribute to float above the *OccupiedCooling* setpoint (i.e., *OccupiedCooling* + *OccupiedSetback*) or below the  
9289 *OccupiedHeating* setpoint (i.e., *OccupiedHeating* – *OccupiedSetback*) before initiating a state change to bring the  
9290 temperature back to the user’s desired setpoint. This attribute is sometimes also referred to as the “span.”

9291 The purpose of this attribute is to allow remote configuration of the span between the desired setpoint and the measured  
9292 temperature to help prevent over-cycling and reduce energy bills, though this may result in lower comfort on the part  
9293 of some users.

9294 A value of 0xff indicates the attribute is unused.

9295 If *OccupiedSetback* is implemented, then the Thermostat server SHALL also implement *OccupiedSetbackMin* and  
9296 *OccupiedSetbackMax* attributes.

9297 If the Thermostat client attempts to write *OccupiedSetback* to a value greater than *OccupiedSetbackMax*, the  
9298 Thermostat server SHALL set its *OccupiedSetback* value to *OccupiedSetbackMax* and SHALL send a Write Attribute  
9299 Response command with a Status Code field enumeration of LIMIT\_REACHED response.

9300 If the Thermostat client attempts to write *OccupiedSetback* to a value less than *OccupiedSetbackMin*, the Thermostat  
9301 server SHALL set its *OccupiedSetback* value to *OccupiedSetbackMin* and SHALL send a Write Attribute Response  
9302 command with a Status Code field enumeration of LIMIT\_REACHED response.

#### 9303 **6.3.2.2.4.5 OccupiedSetbackMin Attribute**

9304 Specifies the minimum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the  
9305 *OccupiedSetback* attribute to be configured by a user.

9306 A value of 0xff indicates the attribute is unused.

9307 *OccupiedSetbackMin* attribute value SHALL be less than *OccupiedSetbackMax* attribute value. Attempts to configure  
9308 *OccupiedSetbackMin* with a value greater than or equal to the value of the *OccupiedSetbackMax* attribute SHALL  
9309 cause the Thermostat server to respond with a Write Attribute Response Command containing the status  
9310 INVALID\_VALUE and SHALL revert back to the previous *OccupiedSetbackMin* attribute value.

9311 If *OccupiedSetbackMin* is configured to a value greater than the value of *OccupiedSetback*, then the Thermostat server  
9312 SHALL update the value of *OccupiedSetback* to equal the new value of *OccupiedSetbackMin*.

#### 9313 **6.3.2.2.4.6 OccupiedSetbackMax Attribute**

9314 Specifies the maximum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the  
9315 *OccupiedSetback* attribute to be configured by a user.

9316 A value of 0xff indicates the attribute is unused.

9317 *OccupiedSetbackMax* attribute value SHALL be greater than *OccupiedSetbackMin* attribute value. Attempts to  
9318 configure *OccupiedSetbackMax* with a value less than or equal to the value of the *OccupiedSetbackMin* attribute  
9319 SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status  
9320 INVALID\_VALUE and SHALL revert back to the previous *OccupiedSetbackMax* attribute value.

9321 If *OccupiedSetbackMax* is configured to a value less than the value of *OccupiedSetback*, then the Thermostat server  
9322 SHALL update the value of *OccupiedSetback* to equal the new value of *OccupiedSetbackMax*.

#### 9323 **6.3.2.2.4.7 UnoccupiedSetback Attribute**

9324 Specifies the degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the *LocalTemperature*  
9325 attribute to float above the *UnoccupiedCooling* setpoint (i.e., *UnoccupiedCooling* + *UnoccupiedSetback*) or below the  
9326 *UnoccupiedHeating* setpoint (i.e., *UnoccupiedHeating* - *UnoccupiedSetback*) before initiating a state change to bring  
9327 the temperature back to the user’s desired setpoint. This attribute is sometimes also referred to as the “span.”

9328 The purpose of this attribute is to allow remote configuration of the span between the desired setpoint and the measured  
9329 temperature to help prevent over-cycling and reduce energy bills, though this may result in lower comfort on the part  
9330 of some users.

- 9331 A value of 0xff indicates the attribute is unused.
- 9332 If UnoccupiedSetback is implemented, then the Thermostat server SHALL also implement UnoccupiedSetbackMin  
9333 and UnoccupiedSetbackMax attributes.
- 9334 If the Thermostat client attempts to write UnoccupiedSetback to a value greater than UnoccupiedSetbackMax, the  
9335 Thermostat server SHALL set its UnoccupiedSetback value to UnoccupiedSetbackMax and SHALL send a Write  
9336 Attribute Response command with a Status Code field enumeration of LIMIT\_REACHED response.
- 9337 If the Thermostat client attempts to write UnoccupiedSetback to a value less than UnoccupiedSetbackMin, the  
9338 Thermostat server SHALL set its UnoccupiedSetback value to UnoccupiedSetbackMin and SHALL send a Write  
9339 Attribute Response command with a Status Code field enumeration of LIMIT\_REACHED response.

#### 9340 **6.3.2.2.4.8 UnoccupiedSetbackMin Attribute**

- 9341 Specifies the minimum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the  
9342 *UnoccupiedSetback* attribute to be configured by a user.
- 9343 A value of 0xff indicates the attribute is unused.
- 9344 *UnoccupiedSetbackMin* attribute value SHALL be less than *UnoccupiedSetbackMax* attribute value. Attempts to  
9345 configure *UnoccupiedSetbackMin* with a value greater than or equal to the value of the *UnoccupiedSetbackMax*  
9346 attribute SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the  
9347 status INVALID\_VALUE and SHALL revert back to the previous *UnoccupiedSetbackMin* attribute value.
- 9348 If *UnoccupiedSetbackMin* is configured to a value greater than the value of *UnoccupiedSetback*, then the Thermostat  
9349 server SHALL update the value of *UnoccupiedSetback* to equal the new value of *UnoccupiedSetbackMin*.

#### 9350 **6.3.2.2.4.9 UnoccupiedSetbackMax Attribute**

- 9351 Specifies the maximum degrees Celsius, in 0.1 degree increments, the Thermostat server will allow the  
9352 *UnoccupiedSetback* attribute to be configured by a user.
- 9353 A value of 0xff indicates the attribute is unused.
- 9354 *UnoccupiedSetbackMax* attribute value SHALL be greater than *UnoccupiedSetbackMin* attribute value. Attempts to  
9355 configure *UnoccupiedSetbackMax* with a value less than or equal to the value of the *UnoccupiedSetbackMin* attribute  
9356 SHALL cause the Thermostat server to respond with a Write Attribute Response Command containing the status  
9357 INVALID\_VALUE and SHALL revert back to the previous *UnoccupiedSetbackMax* attribute value.
- 9358 If *UnoccupiedSetbackMax* is configured to a value less than the value of *UnoccupiedSetback*, then the Thermostat  
9359 server SHALL update the value of *UnoccupiedSetback* to equal the new value of *UnoccupiedSetbackMax*.

#### 9360 **6.3.2.2.4.10 EmergencyHeatDelta Attribute**

- 9361 Specifies the delta, in 0.1 degrees Celsius, between *LocalTemperature* and the *OccupiedHeatingSetpoint* or  
9362 *UnoccupiedHeatingSetpoint* attributes at which the Thermostat server will operate in emergency heat mode.
- 9363 If the difference between *LocalTemperature* and *Un/OccupiedHeatingSetpoint* is greater than or equal to the  
9364 *EmergencyHeatDelta* and the Thermostat server's *SystemMode* attribute is in a heating-related mode, then the  
9365 Thermostat server SHALL immediately switch to the *SystemMode* attribute value that provides the highest stage of  
9366 heating (e.g., emergency heat) and continue operating in that running state until the *OccupiedHeatingSetpoint* value  
9367 is reached. For example:
- 9368     • *LocalTemperature* = 10.0 degrees Celsius
- 9369     • *OccupiedHeatingSetpoint* = 16.0 degrees Celsius
- 9370     • *EmergencyHeatDelta* = 2.0 degrees Celsius
- 9371 =>  $OccupiedHeatingSetpoint - LocalTemperature \geq EmergencyHeatDelta$
- 9372     =>  $16 - 10 \geq 2$
- 9373 => TRUE >>> Thermostat server changes its *SystemMode* to operate in 2<sup>nd</sup> stage or emergency heat mode

9374 The purpose of this attribute is to provide Thermostat clients the ability to configure rapid heating when a setpoint is  
 9375 of a specified amount greater than the measured temperature. This allows the heated space to be quickly heated to the  
 9376 desired level set by the user.

9377 **6.3.2.2.5 AC Information Attribute Set**

9378 **Table 6-28. Attributes of the AC Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0040	<i>ACType</i>	enum8	0x00 – 0x04	RW	0x00	O
0x0041	<i>ACCapacity</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0042	<i>ACRefrigerantType</i>	enum8	0x00-0x03	RW	0x00	O
0x0043	<i>ACCompressorType</i>	enum8	0x00-0x03	RW	0x00	O
0x0044	<i>ACErrorCode</i>	map32	0x00000000 – 0xffffffff	RW	0x00000000	O
0x0045	<i>ACLouverPosition</i>	enum8	0x00 – 0x05	RW	0x00	O
0x0046	<i>ACCoilTemperature</i>	int16	0x954d – 0x7fff	R	-	O
0x0047	<i>ACCapacityFormat</i>	enum8	0x00 – 0xff	RW	0x00	O

9379 **6.3.2.2.5.1 ACType Attribute**

9380 Indicates the type of Mini Split *ACType* of Mini Split AC is defined depending on how Cooling and Heating condition  
 9381 is achieved by Mini Split AC.

9382 **Table 6-29. ACType Enumeration**

Enumeration Field Value	Description
0x00	Reserved
0x01	Cooling and Fixed Speed
0x02	Heat Pump and Fixed Speed
0x03	Cooling and Inverter
0x04	Heat Pump and Inverter

9383 **6.3.2.2.5.2 ACCapacity Attribute**

9384 Indicates capacity of Mini Split AC in terms of the format defined by the *ACCapacityFormat* attribute

9385 **6.3.2.2.5.3 ACRefrigerantType Attribute**

9386 Indicates type of refrigerant used within the Mini Split AC.

9387 **Table 6-30. ACRefrigerantType Enumeration**

Enumeration Field Value	Description
0x00	Reserved

Enumeration Field Value	Description
0x01	R22
0x02	R410a
0x03	R407c

9388

9389 **6.3.2.2.5.4 ACCompressorType Attribute**

9390 This indicates type of Compressor used within the Mini Split AC.

9391

**Table 6-31. ACCompressorType Enumeration**

Enumeration Field Value	Description
0x00	Reserved
0x01	T1, Max working ambient 43 °C
0x02	T2, Max working ambient 35 °C
0x03	T3, Max working ambient 52 °C

9392 **6.3.2.2.5.5 ACErrorCode Attribute**9393 This indicates the type of errors encountered within the Mini Split AC. Error values are reported with four bytes  
9394 values. Each bit within the four bytes indicates the unique error.

9395

**Table 6-32. ACErrorCode Values**

Bit	Value
0	Compressor Failure or Refrigerant Leakage
1	Room Temperature Sensor Failure
2	Outdoor Temperature Sensor Failure
3	Indoor Coil Temperature Sensor Failure
4	Fan Failure

9396 **6.3.2.2.5.6 ACLouverPosition Attribute**

9397 This indicates the position of Louver on the AC. Attribute values are listed in Table 6-33.

9398

**Table 6-33. ACLouverPosition Values**

Louver Position Byte	Value
Fully Closed	0x01
Fully Open	0x02
Quarter Open	0x03
Half Open	0x04

Louver Position Byte	Value
Three Quarters Open	0x05

9399 **6.3.2.2.5.7 ACCoilTemperature Attribute**

9400 *ACCoilTemperature* represents the temperature in degrees Celsius, as measured locally or remotely (over the network)  
 9401 as follows:

- 9402 • *ACCoilTemperature* = 100 x temperature in degrees Celsius.
- 9403 • Where  $-273.15^{\circ}\text{C} \leq \text{temperature} \leq 327.67^{\circ}\text{C}$ , corresponding to an *ACCoilTemperature* in the range 0x954d  
 9404 to 0x7fff.
- 9405 • The maximum resolution this format allows is 0.01 °C.
- 9406 • *ACCoilTemperature* of 0x8000 indicates that the temperature measurement is invalid.

9407 **6.3.2.2.5.8 ACCapacityFormat Attribute**

9408 This is the format for the *ACCapacity* attribute.

9409 **Table 6-34. ACCapacity Enumeration**

Enumeration Field Value	Description
0x00	BTUh

9410 **6.3.2.3 Server Commands Received**

9411 The command IDs for the Thermostat cluster are listed in Table 6-35:

9412 **Table 6-35. Command IDs for the Thermostat Cluster**

Command Identifier Field Value	Description	M/O
0x00	Setpoint Raise/Lower	M
0x01	Set Weekly Schedule	O
0x02	Get Weekly Schedule	O
0x03	Clear Weekly Schedule	O
0x04	Get Relay Status Log	O

9413 **6.3.2.3.1 Setpoint Raise/Lower Command**

9414 **6.3.2.3.1.1 Payload Format**

9415 The Setpoint Raise/Lower command payload SHALL be formatted as illustrated in Figure 6-4Format of the Setpoint  
 9416 Raise/Lower Command Payload.

9417

**Figure 6-4. Format of the Setpoint Raise/Lower Command Payload**

<b>Bits</b>	8	8
<b>Data Type</b>	enum8	int8
<b>Field Name</b>	Mode	Amount

9418 **6.3.2.3.1.2 Mode Field**

9419 The mode field SHALL be set to one of the non-reserved values in Table 6-36. It specifies which setpoint is to be  
9420 configured. If it is set to auto, then both setpoints SHALL be adjusted.

9421 **Table 6-36. Mode Field Values for Setpoint Raise/Lower Command**

Mode Field Value	Description
0x00	Heat (adjust Heat Setpoint)
0x01	Cool (adjust Cool Setpoint)
0x02	Both (adjust Heat Setpoint and Cool Setpoint)

9422 **6.3.2.3.1.3 Amount Field**

9423 The amount field is a signed 8-bit integer that specifies the amount the setpoint(s) are to be increased (or decreased)  
9424 by, in steps of 0.1°C.

9425 **6.3.2.3.1.4 Effect on Receipt**

9426 The attributes for the indicated setpoint(s) SHALL be increased by the amount specified in the Amount field.

9427 **6.3.2.3.2 Set Weekly Schedule Command**9428 **6.3.2.3.2.1 Payload Format**

9429 The set weekly schedule command payload SHALL be formatted as shown in Figure 6-5 and Figure 6-6.

9430 **Figure 6-5. Set Weekly Schedule Command Payload Format (1 of 2)**

<b>Octets</b>	1(Header)	1(Header)	1(Header)	2	2/0	2/0
<b>Data Type</b>	enum8	map8	map8	uint16	int16	int16
<b>Field Name</b>	Number of Transitions for Sequence	Day of Week for Sequence	Mode for Sequence	Transition Time 1	Heat Set Point 1	Cool Set Point 1

9431

9432

**Figure 6-6. Set Weekly Schedule Command Payload Format (2 of 2)**

<b>Octets</b>	Variable	2	2/0	2/0
<b>Data Type</b>	--	uint16	int16	int16
<b>Field Name</b>	--	Transition Time 10	Heat Set Point 10	Cool Set Point 10



9433 The set weekly schedule command is used to update the thermostat weekly set point schedule from a management  
 9434 system. If the thermostat already has a weekly set point schedule programmed then it SHOULD replace each daily set  
 9435 point set as it receives the updates from the management system. For example if the thermostat has 4 set points for  
 9436 every day of the week and is sent a Set Weekly Schedule command with one set point for Saturday then the thermostat  
 9437 SHOULD remove all 4 set points for Saturday and replace those with the updated set point but leave all other days  
 9438 unchanged. If the schedule is larger than what fits in one frame or contains more than 10 transitions, the schedule  
 9439 SHALL then be sent using multiple Set Weekly Schedule Commands.

9440 Each Set Weekly Schedule Command has 3 header bytes – Number of Transitions for Sequence, Day of Week for  
 9441 Sequence, and Mode for Sequence. The application SHALL decode the payload according to what has specified in  
 9442 the 3 header bytes.

9443 **6.3.2.3.2.2 Number of Transitions for Sequence**

9444 The Number of Transitions for Sequence field indicates how many individual transitions to expect for this sequence  
 9445 of commands. If a device supports more than 10 transitions in its schedule they can send this by sending more than 1  
 9446 “Set Weekly Schedule” command, each containing the separate information that the device needs to set.

9447 **6.3.2.3.2.3 Day of Week for Sequence**

9448 This field represents the day of the week at which all the transitions within the payload of the command SHOULD be  
 9449 associated to. This field is a bitmap and therefore the associated set point could overlap onto multiple days (you could  
 9450 set one transition time for all “week days” or whatever combination of days the implementation requests). Table 6-37  
 9451 displays the bitmap values.

9452 **Table 6-37. Day Of Week for Sequence Values**

Bit Number	Description
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Away or Vacation

9453  
 9454 Each set point transition will begin with the day of week for this transition. There can be up to 10 transitions for each  
 9455 command.

9456 **6.3.2.3.2.4 Mode for Sequence**

9457 This field determines how the application SHALL decode the Set Point Fields of each transition for the remaining of  
 9458 the command. This field is a bitmap and the values are presented in Table 6-38.

9459 **Table 6-38. Mode for Sequence Values**

Bit Number	Description
0	Heat Setpoint Field Present in Payload

Bit Number	Description
1	Cool Setpoint Field Present in Payload

9460 If the Heat Bit is On and the Cool Bit is Off, the Command SHALL be represented as in Figure 6-7 and Figure 6-8.

9461 **Figure 6-7. Set Heat Weekly Schedule Command Payload Format (1 of 2)**

Octets	1(Header)	1(Header)	1(Header)	2	2
Data Type	enum8	map8	map8	uint16	int16
Field Name	Number of Transitions for Sequence	Day of Week for Sequence	0x01 (Heat)	Transition Time 1	Heat Set Point 1

9462 **Figure 6-8. Set Heat Weekly Schedule Command Payload Format (2 of 2)**

Octets	Variable	2	2
Data Type	--	uint16	int16
Field Name	--	Transition Time 10	Heat Set Point 10

9464 If the Heat Bit is Off and the Cool Bit is On, the Command SHALL be represented as in Figure 6-9 and Figure 6-10.

9466 **Figure 6-9. Set Cool Weekly Schedule Command Payload Format (1 of 2)**

Octets	1(Header)	1(Header)	1(Header)	2	2
Data Type	enum8	map8	map8	uint16	int16
Field Name	Number of Transitions for Sequence	Day of Week for Sequence	0x02 (Cool)	Transition Time 1	Cool Set Point 1

9467 **Figure 6-10. Set Cool Weekly Schedule Command Payload Format (2 of 2)**

Octets	Variable	2	2
Data Type	--	uint16	int16
Field Name	--	Transition Time 10	Cool Set Point 10

9469 If both the Heat Bit and the Cool Bit are On, the Command SHALL be represented as in Figure 6-11 and Figure 6-12.

9471 **Figure 6-11. Set Heat & Cool Weekly Schedule Command Payload Format (1 of 2)**

Octets	1(Header)	1(Header)	1(Header)	2	2	2
Data Type	enum8	map8	map8	uint16	int16	int16

<b>Field Name</b>	Number of Transitions for Sequence	Day of Week for Sequence	0x03 (Heat & Cool)	Transition Time 1	Heat Set Point 1	Cool Set Point 1
-------------------	------------------------------------	--------------------------	--------------------	-------------------	------------------	------------------

9472  
 9473

**Figure 6-12. Set Heat & Cool Weekly Schedule Command Payload Format (2 of 2)**

<b>Octets</b>	Variable	2	2	2
<b>Data Type</b>	--	uint16	int16	int16
<b>Field Name</b>	--	Transition Time 10	Heat Set Point 10	Cool Set Point 10

9474

9475 At least one of the bits in the Mode For Sequence byte SHALL be on.

9476 **6.3.2.3.2.5 Transition Time Field**

9477 This field represents the start time of the schedule transition during the associated day. The time will be represented  
 9478 by a 16 bits unsigned integer to designate the minutes since midnight. For example, 6am will be represented by 0x0168  
 9479 (360 minutes since midnight) and 11:30pm will be represented by 0x0582 (1410 minutes since midnight)

9480 **6.3.2.3.2.6 Heat Set Point Field**

9481 If the heat bit is enabled in the *Mode For Sequence* byte, this field represents the heat setpoint to be applied at this  
 9482 associated transition start time. The format of this attribute represents the temperature in degrees Celsius with 0.01  
 9483 deg C resolution.

9484 **6.3.2.3.2.7 Cool Set Point Field**

9485 If the cool bit is enabled in the *Mode For Sequence* byte, this field represents the cool setpoint to be applied at this  
 9486 associated transition start time. The format of this attribute represents the temperature in degrees Celsius with 0.01  
 9487 deg C resolution.

9488 **6.3.2.3.2.8 Effect on Receipt**

9489 The weekly schedule for updating set points SHALL be stored in the thermostat and SHOULD begin at the time of  
 9490 receipt. A default response SHALL always be sent as a response. If the total number of transitions sent is greater than  
 9491 what the thermostat supports a default response of INSUFFICIENT\_SPACE (0x89) SHALL be sent in response to  
 9492 the last command sent for that transition sequence. If any of the set points sent in the entire sequence is out of range  
 9493 of what the thermostat supports (AbsMin/MaxSetPointLimit) then a default response of INVALID\_VALUE (0x87)  
 9494 SHALL be sent in return and the no set points from the entire sequence SHOULD be used. If the transitions could be  
 9495 added successfully a default response of SUCCESS(0x00) SHALL be sent. Overlapping transitions is not allowed. If  
 9496 an overlap is detected and a default response of FAILURE(0x01) SHALL be sent. The Day of Week for Sequence  
 9497 and Mode for Sequence fields are defined as bitmask for the flexibility to support multiple days and multiple modes  
 9498 within one command. If thermostat cannot handle incoming command with multiple days and/or multiple modes  
 9499 within one command, it SHALL send default response of INVALID\_FIELD (0x85) in return.

9500 **6.3.2.3.3 Get Weekly Schedule**9501 **Figure 6-13. Format of the Get Weekly Schedule Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	map8	map8
<b>Field Name</b>	Days To Return	Mode To Return

9502 **6.3.2.3.3.1 Days To Return**

9503 This field indicates the number of days the client would like to return the set point values for and could be any  
9504 combination of single days or the entire week. This field has the same format as the Day of Week for Sequence field  
9505 in the **Set Weekly Schedule command**.

9506 **6.3.2.3.3.2 Mode To Return**

9507 This field indicates the mode the client would like to return the set point values for and could be any combination of  
9508 heat only, cool only or heat&cool. This field has the same format as the Mode for Sequence field in the **Set Weekly**  
9509 **Schedule command**.

9510 **6.3.2.3.3.3 Effect on Receipt**

9511 When this command is received the unit SHOULD send in return the Get Weekly Schedule Response<sup>112</sup> command.  
9512 The Days to Return and Mode to Return fields are defined as bitmask for the flexibility to support multiple days and  
9513 multiple modes within one command. If thermostat cannot handle incoming command with multiple days and/or  
9514 multiple modes within one command, it SHALL send default response of INVALID\_FIELD (0x85) in return.

9515 **6.3.2.3.4 Clear Weekly Schedule**

9516 The Clear Weekly Schedule command is used to clear the weekly schedule. The Clear weekly schedule has no payload.

9517 **6.3.2.3.4.1 Effect on Receipt**

9518 When this command is received, all transitions currently stored SHALL be cleared and a default response of  
9519 SUCCESS (0x00) SHALL be sent in response. There are no error responses to this command.

9520 **6.3.2.3.5 Get Relay Status Log**

9521 The Get Relay Status Log command is used to query the thermostat internal relay status log. This command has no  
9522 payload.

9523 The log storing order is First in First Out (FIFO) when the log is generated and stored into the Queue.

9524 The first record in the log (i.e., the oldest) one, is the first to be replaced when there is a new record and  
9525 there is no more space in the log. Thus, the newest record will overwrite the oldest one if there is no space  
9526 left.

9527 The log storing order is Last In First Out (LIFO) when the log is being retrieved from the Queue by a client  
9528 device.

9529 Once the "Get Relay Status Log Response" frame is sent by the Server, the "Unread Entries" attribute  
9530 SHOULD be decremented to indicate the number of unread records that remain in the queue.

<sup>112</sup> CCB 2251 the command is Get Weekly Schedule Response (not Current Weekly Schedule)

9531 If the "Unread Entries" attribute reaches zero and the Client sends a new "Get Relay Status Log Request",  
 9532 the Server MAY send one of the following items as a response:

9533 i) resend the last Get Relay Status Log Response

9534 or

9535 ii) generate new log record at the time of request and send Get Relay Status Log Response with  
 9536 the new data

9537 For both cases, the "Unread Entries" attribute will remain zero.

9538

9539 **6.3.2.3.5.1 Effect on Receipt**

9540 When this command is received, the unit SHALL respond with Relay Status Log command if the relay status log  
 9541 feature is supported on the unit.

9542 **6.3.2.4 Server Commands Sent**

9543 Table 6-39 shows the command sent by the server (received by the client).

9544 **Table 6-39. Server Commands Send Command ID**

Command Identifier Field Value	Description
0x00	Get Weekly Schedule Response
0x01	Get Relay Status Log Response

9545 **6.3.2.4.1 Get Weekly Schedule Response**

9546 This command has the same payload format as the Set Weekly Schedule. Please refer to the payload detail in Section  
 9547 Set Weekly Schedule Command, Set Weekly Schedule Command, in this chapter.

9548 **6.3.2.4.2 Get Relay Status Log Response**

9549 This command is sent from the thermostat cluster server in response to the Get Relay Status Log. After the Relay  
 9550 Status Entry is sent over the air to the requesting client, the specific entry will be cleared from the thermostat internal  
 9551 log.

9552 **6.3.2.4.2.1 Payload Format**

9553 The relay status log command payload SHALL be formatted as shown in Figure 6-14.

9554

Figure 6-14. Format of the Relay Status Log Payload

<b>Octets</b>	2	2	2	1	2	2
<b>Data Type</b>	uint16	map8	int16	uint8	int16	uint16
<b>Field Name</b>	Time of Day	Relay Status	Local Temperature	Humidity in Percentage	Set Point	Unread Entries

9555 **6.3.2.4.2.2 Time of Day Field**

9556 Represents the sample time of the day, in minutes since midnight, when the relay status was captured for this associated  
9557 log entry. For example, 6am will be represented by 0x0168 (360 minutes since midnight) and 11:30pm will be  
9558 represented by 0x0582 (1410 minutes since midnight).

9559 **6.3.2.4.2.3 Relay Status Field**

9560 Presents the relay status for thermostat when the log is captured. Each bit represents one relay used by the thermostat.  
9561 If the bit is on, the associated relay is on and active. Each thermostat manufacturer can create its own mapping between  
9562 the bitmask and the associated relay.

9563 **6.3.2.4.2.4 Local Temperature Field**

9564 Presents the local temperature when the log is captured. The format of this attribute represents the temperature in  
9565 degrees Celsius with 0.01 deg C resolution.

9566 **6.3.2.4.2.5 Humidity Field**

9567 This field presents the humidity as a percentage when the log was captured.

9568 **6.3.2.4.2.6 Setpoint Field**

9569 Presents the target setpoint temperature when the log is captured. The format of this attribute represents the  
9570 temperature in degrees Celsius with 0.01 deg C resolution.

9571 **6.3.2.4.2.7 Unread Entries Field**

9572 This field presents the number of unread entries within the thermostat internal log system.

9573 **6.3.2.5 Attribute Reporting**

9574 This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum  
9575 and maximum reporting interval and reportable change settings described in Chapter 2, Foundation and whenever  
9576 they change. The following attributes SHALL be reported:

- 9577 • *LocalTemperature*
- 9578 • *PICoolingDemand*
- 9579 • *PIHeatingDemand*

9580 Other attributes MAY optionally be reported.

9581 **6.3.2.6 Scene Table Extensions**

9582 If the Scenes server cluster (see 3.7) is implemented, the following extension fields SHALL be added to the Scenes  
9583 table in the given order, i.e., the attribute listed as 1 is added first:

- 9584 1) *OccupiedCoolingSetpoint*

9585 2) *OccupiedHeatingSetpoint*

9586 3) *SystemMode*

### 9587 6.3.3 Client

9588 The client has no specific dependencies nor specific attributes. The client cluster generates the commands received by  
 9589 the server cluster, as required by the application.

## 9590 6.4 Fan Control

### 9591 6.4.1 Overview

9592 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 9593 etc.

9594 This cluster specifies an interface to control the speed of a fan as part of a heating / cooling system.

#### 9595 6.4.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

#### 9596 6.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	FAN	Type 1 (client to server)

#### 9597 6.4.1.3 Cluster Identifiers

Identifier	Name
0x0202	Fan Control

### 9598 6.4.2 Server

#### 9599 6.4.2.1 Attributes

9600 The Fan Control Status attribute set contains the attributes summarized in Table 6-40Attributes of the Fan Control  
 9601 Cluster.

9602 **Table 6-40. Attributes of the Fan Control Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>FanMode</i>	enum8	0x00 – 0x06	RW	0x05 (auto)	M
0x0001	<i>FanModeSequence</i>	enum8	0x00 – 0x04	RW	0x02	M

9603 **6.4.2.1.1 FanMode Attribute**9604 The *FanMode* attribute is an 8-bit value that specifies the current speed of the fan. It SHALL be set to one of the  
9605 nonreserved values in Table 6-41:

9606

9607

Table 6-41. *FanMode* Attribute Values

Value	Description
0x00	Off
0x01	Low
0x02	Medium
0x03	High
0x04	On
0x05	Auto (the fan speed is self-regulated)
0x06	Smart (when the heated/cooled space is occupied, the fan is always on)

9608 Note that for Smart mode, information must be available as to whether the heated/cooled space is occupied. This MAY  
9609 be accomplished by use of the Occupancy Sensing cluster (see 4.8).9610 **6.4.2.1.2 FanModeSequence Attribute**9611 The *FanModeSequence* attribute is an 8-bit value that specifies the possible fan speeds that the thermostat can set. It  
9612 SHALL be set to one of the non-reserved values in Table 6-42*FanSequenceOperatio*. (**Note:** 'Smart' is not in this  
9613 table, as this mode resolves to one of the other modes depending on occupancy).

9614

Table 6-42. *FanSequenceOperation* Attribute Values

Attribute Value	Description
0x00	Low/Med/High
0x01	Low/High
0x02	Low/Med/High/Auto
0x03	Low/High/Auto
0x04	On/Auto

9615 **6.4.2.2 Commands**

9616 No cluster specific commands are generated or received by the server.

9617 **6.4.3 Client**9618 The Client cluster has no specific attributes. No cluster specific commands are received by the server. No cluster  
9619 specific commands are generated by the server.



## 9620 6.5 Dehumidification Control

### 9621 6.5.1 Overview

9622 This cluster provides an interface to dehumidification functionality.

#### 9623 6.5.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

#### 9624 6.5.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DHUM	Type 1 (client to server)

#### 9625 6.5.1.3 Cluster Identifiers

Identifier	Name
0x0203	Dehumidification Control

## 9626 6.5.2 Server

### 9627 6.5.2.1 Attributes

9628 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 9629 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant nibble specifies the attribute  
 9630 set and the least significant nibble specifies the attribute within the set. The currently defined attribute set for the  
 9631 dehumidification control cluster is listed in Table 6-43.

9632 **Table 6-43. Dehumidification Control Attribute Sets**

Attribute Set Identifier	Description
0x000	Dehumidification Information
0x001	Dehumidification Settings

#### 9633 6.5.2.1.1 Dehumidification Information Attribute Set

9634 The Dehumidification Information attribute set contains the attributes summarized in Table 6-44  
 9635 Dehumidification Information Attribute Set.

9636

**Table 6-44. Dehumidification Information Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0000	<i>RelativeHumidity</i>	uint8	0x00 – 0x64	R	-	O
0x0001	<i>DehumidificationCooling</i>	uint8	0 - <i>DehumidificationMaxCool</i>	RP	-	M

9637 **6.5.2.1.1.1 *RelativeHumidity* Attribute**

9638 The *RelativeHumidity* attribute is an 8-bit value that represents the current relative humidity (in %) measured by a  
9639 local or remote sensor. The valid range is 0x00 – 0x64 (0% to 100%).

9640 **6.5.2.1.1.2 *DehumidificationCooling* Attribute**

9641 The *DehumidificationCooling* attribute is an 8-bit value that specifies the current dehumidification cooling output (in  
9642 %). The valid range is 0 to *DehumidificationMaxCool*.

9643 **6.5.2.1.2 Dehumidification Settings Attribute Set**

9644 The Dehumidification Settings attribute set contains the attributes summarized in the table below:

9645

**Table 6-45. Dehumidification Settings Attribute Set**

<b>Identifier</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Default</b>	<b>M/O</b>
0x0010	<i>RHDehumidificationSetpoint</i>	uint8	0x1E – 0x64	RW	0x32	M
0x0011	<i>RelativeHumidityMode</i>	enum8	0x00 – 0x01	RW	0x00	O
0x0012	<i>DehumidificationLockout</i>	enum8	0x00 – 0x01	RW	0x01	O
0x0013	<i>DehumidificationHysteresis</i>	uint8	0x02 – 0x14	RW	0x02	M
0x0014	<i>DehumidificationMaxCool</i>	uint8	0x14 – 0x64	RW	0x14	M
0x0015	<i>RelativeHumidityDisplay</i>	enum8	0x00 – 0x01	RW	0x00	O

9646 **6.5.2.1.2.1 *RHDehumidificationSetpoint* Attribute**

9647 The *RHDehumidificationSetpoint* attribute is an 8-bit value that represents the relative humidity (in %) at which  
9648 dehumidification occurs. The valid range is 0x1E – 0x64 (30% to 100%).

9649 **6.5.2.1.2.2 *RelativeHumidityMode* Attribute**

9650 The *RelativeHumidityMode* attribute is an 8-bit value that specifies how the *RelativeHumidity* value is being updated.  
9651 It SHALL be set to one of the values below:

9652

**Table 6-46. *RelativeHumidityMode* Attribute Values**

<b>Attribute Value</b>	<b>Description</b>
0x00	<i>RelativeHumidity</i> measured locally
0x01	<i>RelativeHumidity</i> updated over the network

9653 **6.5.2.1.2.3 *DehumidificationLockout* Attribute**

9654 The *DehumidificationLockout* attribute is an 8-bit value that specifies whether dehumidification is allowed or not. It  
 9655 SHALL be set to one of the values below:

9656 **Table 6-47. *DehumidificationLockout* Attribute Values**

Attribute Value	Description
0x00	Dehumidification is not allowed.
0x01	Dehumidification is allowed.

9657 **6.5.2.1.2.4 *DehumidificationHysteresis* Attribute**

9658 The *DehumidificationHysteresis* attribute is an 8-bit value that specifies the hysteresis (in %) associated with  
 9659 *RelativeHumidity* value. The valid range is 0x02 – 0x14 (2% to 20%).

9660 **6.5.2.1.2.5 *DehumidificationMaxCool* Attribute**

9661 The *DehumidificationMaxCool* attribute is an 8-bit value that specifies the maximum dehumidification cooling output  
 9662 (in %). The valid range is 0x14 – 0x64 (20% to 100%).

9663 **6.5.2.1.2.6 *RelativeHumidityDisplay* Attribute**

9664 The *RelativeHumidityDisplay* attribute is an 8-bit value that specifies whether the *RelativeHumidity* value is displayed  
 9665 to the user or not. It SHALL be set to one of the non-reserved values in Table 6-48.

9666 **Table 6-48. *RelativeHumidityMode* Attribute Values**

Attribute Value	Description
0x00	<i>RelativeHumidity</i> is not displayed
0x01	<i>RelativeHumidity</i> is displayed

9667 **6.5.2.2 Commands**

9668 No cluster specific commands are generated or received by the server.

9669 **6.5.2.3 Attribute Reporting**

9670 This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum  
 9671 and maximum reporting interval settings described in the ZCL Foundation specification.

9672 The following attribute SHALL be reported: *DehumidificationCooling*

9673 This attribute SHALL also be reported whenever it changes (a minimum change is 1%).

9674 Reports of this attribute MAY be used to control a remote dehumidifier device.

9675 **6.5.3 Client**

9676 The client has no dependencies or attributes and there are no cluster specific commands defined.

9677 

## 6.6 Thermostat User Interface Configuration

9678 

### 6.6.1 Overview

9679 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
9680 etc.9681 This cluster provides an interface to allow configuration of the user interface for a thermostat, or a thermostat controller  
9682 device, that supports a keypad and LCD screen.9683 

#### 6.6.1.1 Revision History

Rev	Description
1	global mandatory <i>ClusterRevision</i> attribute added

9684 

#### 6.6.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TSUIC	Type 1 (client to server)

9685 

#### 6.6.1.3 Cluster Identifiers

Identifier	Name
0x0204	Thermostat User Interface Configuration

9686 

### 6.6.2 Server

9687 

#### 6.6.2.1 Attributes

9688 The attributes of this cluster are summarized in Table 6-49.

9689 **Table 6-49. Thermostat User Interface Configuration Cluster**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>TemperatureDisplayMode</i>	enum8	0x00 – 0x01	RW	0x00 (Celsius)	M
0x0001	<i>KeypadLockout</i>	enum8	0x00 – 0x05	RW	0x00 (no lockout)	M
0x0002	<i>ScheduleProgrammingVisibility</i>	enum8	0x00 – 0x01	RW	0x00	O

9690 

##### 6.6.2.1.1 *TemperatureDisplayMode* Attribute

9691 The *TemperatureDisplayMode* attribute specifies the units of the temperature displayed on the thermostat screen. This  
9692 attribute SHALL be set to one of the non-reserved values in Table 6-50.

9693

**Table 6-50. *DisplayMode* Attribute Values**

Attribute Value	Description
0x00	Temperature in °C
0x01	Temperature in °F

9694 **6.6.2.1.2 KeypadLockout Attribute**

9695 The *KeypadLockout* attribute specifies the level of functionality that is available to the user via the keypad. This  
 9696 attribute SHALL be set to one of the non-reserved values Table 6-51*KeypadLockou*.

9697

**Table 6-51. *KeypadLockout* Attribute Values**

Attribute Value	Description
0x00	No lockout
0x01	Level 1 lockout
0x02	Level 2 lockout
0x03	Level 3 lockout
0x04	Level 4 lockout
0x05	Level 5 lockout (least functionality available to the user)

9698

9699 The interpretation of the various levels is device-dependent.

9700 **6.6.2.1.3 ScheduleProgrammingVisibility Attribute**

9701 The *ScheduleProgrammingVisibility* attribute is used to hide the weekly schedule programming functionality or menu  
 9702 on a thermostat from a user to prevent local user programming of the weekly schedule. The schedule programming  
 9703 MAY still be performed via a remote interface, and the thermostat MAY operate in schedule programming mode.

9704 This command is designed to prevent local tampering with or disabling of schedules that MAY have been programmed  
 9705 by users or service providers via a more capable remote interface. The programming schedule SHALL continue to run  
 9706 even though it is not visible to the user locally at the thermostat.

9707 It SHALL be set to one of the non-reserved values in Table 6-52.

9708

**Table 6-52. *ScheduleProgrammingVisibility* Attribute Values**

<i>ScheduleProgrammingVisi</i> <i>bility</i> Attribute Value	Description
0x00	Local schedule programming functionality is enabled at the thermostat
0x01	Local schedule programming functionality is disabled at the thermostat

## 9709 6.6.2.2 Commands

9710 No cluster specific commands are generated or received by the server.

## 9711 6.6.2.3 Sample Conversion Code

9712 Sample code provided to ensure consistent Fahrenheit to Celsius and vice-versa conversion between devices and  
9713 across vendors.

9714 For degF: the value is a int8u representing 2x temperature  
9715 value in Farenheit (to get 0.5 resolution).

9716 For degC: the value is a int16s representing Celsius in  
9717 0.01 resolution as expected by the ZCL format.

```
9718 /*  
9719  * Function   : translateZclTemp()  
9720  * Description : Converts the temperature setpoints in ZCL  
9721  *             to the half degF format.  
9722  *             The half degF format is 8-bit unsigned,  
9723  *             and represents 2x temperature value in  
9724  *             Farenheit (to get 0.5 resolution).  
9725  *             The format used in ZCL is 16-bit signed  
9726  *             in Celsius and multiplied by 100  
9727  *             to get 0.01 resolution.  
9728  *             e.g. 2500(25.00 deg C) ---> 0x9A (77 deg F)  
9729  * Input Para : Temperature in ZCL (degC)format  
9730  * Output Para: Temperature in half DegF format  
9731  */
```

```
9732 int8u translateZclTemp(int16s temperature)  
9733 {  
9734     int32s x = temperature;  
9735     //rearrangement of  
9736     // = (x * (9/5) / 100 + 32) * 2;  
9737     // the added 250 is for proper rounding.  
9738     // a rounding technique that only works  
9739     // with positive numbers  
9740  
9741     return (int8u) ((x*9*2 + 250)/ (5*100) + 64);  
9742 }
```

```
9743  
9744 /*  
9745  * Function   : translateDegFTemp  
9746  * Description : Converts the temperature in DegF  
9747  *             protocol to the format  
9748  *             expected by the cluster attribute  
9749  *             Measured Value in the  
9750  *             Temperature Measurement  
9751  *             Information Attribute Set.  
9752  *             The half deg F format is 8-bit  
9753  *             unsigned, and represents  
9754  *             2x temperature value in  
9755  *             Farenheit (to get 0.5 resolution).  
9756  *             The format expected by cluster  
9757  *             is 16-bit signed in Celsius and  
9758  *             multiplied by 100 to get  
9759  *             0.01 resolution.  
9760  *             e.g. 0x9A(77 deg F) ---> 2500 (25.00 deg C)  
9761  * Input Para : temperature in DegF format
```

```
9762 * Output Para: temperature in ZCL format
9763 */
9764 int16s translateDegFTemp(int8u temperature)
9765 {
9766     int32s x = temperature;
9767
9768     // rearrangement of
9769     // = 100 * (x/2 - 32) * 5/9
9770     // *1000 (should be 100), +90, then /10,
9771     // is for rounding.
9772
9773     return (int16s) (((x - 64)*5*1000 + 90) / (10*2*9));
9774 }
```

### 9775 **6.6.3 Client**

9776 The client has no dependencies or cluster specific attributes and there are no cluster specific commands defined.





9777

## CHAPTER 7 CLOSURES

9778  
9779  
9780  
9781

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

9782

### 7.1 General Description

9783

#### 7.1.1 Introduction

9784  
9785

The clusters specified in this document are for use typically in applications involving closures (e.g., shades, windows, doors), but MAY be used in any application domain.

9786

#### 7.1.2 Cluster List

9787  
9788

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

9789

The clusters defined in this document are listed in Table 7-1.

9790

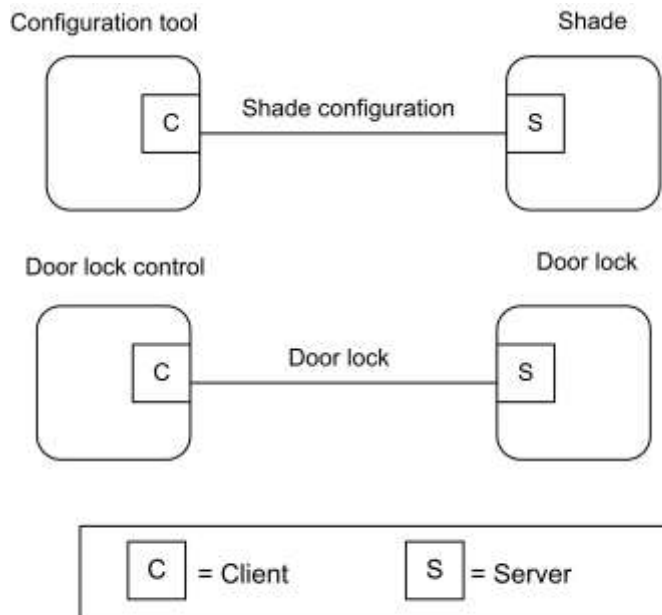
**Table 7-1. Clusters Specified in the Closures Functional Domain**

Cluster ID	Cluster Name	Description
0x0100	Shade Configuration	Attributes and commands for configuring a shade
0x0101	Door Lock	An interface to a generic way to secure a door
0x0102	Window Covering	Commands and attributes for controlling a window covering

9791

9792

Figure 7-1. Typical Usage of the Closures Clusters



9793

*Note: Device names are examples for illustration purposes only*

9794

## 7.2 Shade Configuration

9795

### 7.2.1 Overview

9796  
9797

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

9798

This cluster provides an interface for reading information about a shade, and configuring its open and closed limits.

9799

#### 7.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

9800

#### 7.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SHDCFG	Type 2 (server to client)

9801

#### 7.2.1.3 Cluster Identifiers

Identifier	Name
0x0100	Shade Configuration

9802 **7.2.2 Server**

9803 **7.2.2.1 Attributes**

9804 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 9805 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 9806 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 9807 are listed in Table 7-2.

9808 **Table 7-2. Shade Configuration Attribute Sets**

Attribute Set Identifier	Description
0x000	Shade information
0x001	Shade settings

9809 **7.2.2.1.1 Shade Information Attribute Set**

9810 The Shade Information attribute set contains the attributes summarized in Table 7-3.

9811 **Table 7-3. Attributes of the Shade Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>PhysicalClosedLimit</i>	uint16	0x0001 – 0xfffe	R	-	O
0x0001	<i>MotorStepSize</i>	uint8	0x00 – 0xfe	R	-	O
0x0002	<i>Status</i>	map8	0000 xxxx	RW	0000 0000	M

9812 **7.2.2.1.1.1 PhysicalClosedLimit Attribute**

9813 The *PhysicalClosedLimit* attribute indicates the most closed (numerically lowest) position that the shade can  
 9814 physically move to. This position is measured in terms of steps of the motor, taking the physical most open position  
 9815 of the shade as zero.

9816 This attribute is for installation informational purposes only.

9817 The value 0xffff indicates an invalid or unknown *PhysicalClosedLimit*.

9818 **7.2.2.1.1.2 MotorStepSize Attribute**

9819 The *MotorStepSize* attribute indicates the angle the shade motor moves for one step, measured in 1/10ths of a degree.

9820 This attribute is for installation informational purposes only.

9821 The value 0xff indicates an invalid or unknown step size.

9822 **7.2.2.1.1.3 Status Attribute**

9823 The *Status* attribute indicates the status of a number of shade functions, as shown in Table 7-4. Writing a value to this  
 9824 attribute only affects those bits with Read/Write access.

9825

Table 7-4. Bit Values for the *Status* Attribute

Bit Number	Meaning	Access
0	Shade operational 0 = no 1 = yes	R
1	Shade adjusting 0 = no 1 = yes	R
2	Shade direction 0 = closing 1 = opening	R
3	Direction corresponding to forward direction of motor 0 = closing 1 = opening	RW

9826 **7.2.2.1.2 Shade Settings Attribute Set**

9827 The Shade Settings attribute set contains the attributes summarized in Table 7-5.

9828

Table 7-5. Attributes of the Shade Settings Attribute Set

Id	Name	Type	Range	Access	Default	M/O
0x0010	<i>ClosedLimit</i>	uint16	0x0001 – 0xfffe	RW	0x0001	M
0x0011	<i>Mode</i>	enum8	0x00 – 0xfe	RW	0x00	M

9829 **7.2.2.1.2.1 ClosedLimit Attribute**

9830 The *ClosedLimit* attribute indicates the most closed position that the shade can move to. This position is measured in  
 9831 terms of steps of the motor, taking the physical most open position of the shade as zero. This attribute is set either by  
 9832 directly writing it, or by the following method.

9833 When the *Mode* attribute is set to Configure, the shade is opening, and either the shade is stopped or it reaches its  
 9834 physical most open limit (if there is one – the motor MAY continue to turn at the top), the zero point for the motor-  
 9835 step measurement system is set to the current position of the shade.

9836 When the *Mode* attribute is set to Configure, the shade is closing, and either the shade is stopped or it reaches its  
 9837 physical closed limit, the *ClosedLimit* attribute is set to the current position of the shade, relative to the zero point set  
 9838 as described above.

9839 **7.2.2.1.2.2 Mode Attribute**

9840 The *Mode* attribute indicates the current operating mode of the shade, as shown in Table 7-6. The value 0xff indicates  
 9841 an invalid or unknown mode.

9842

Table 7-6. Values of the Mode Attribute

Attribute Value	Meaning
0x00	Normal
0x01	Configure

9843 In configure mode, the *ClosedLimit* attribute MAY be set as described above.

9844 **7.2.2.2 Commands**

9845 No cluster specific commands are generated or received by the server.

9846 **7.2.3 Client**

9847 The client has no specific attributes and there are no cluster specific commands to receive or generate.

9848 **7.3 Door Lock**

---

9849 **7.3.1 Overview**

9850 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 9851 etc.

9852 The door lock cluster provides an interface to a generic way to secure a door. The physical object that provides the  
 9853 locking functionality is abstracted from the cluster. The cluster has a small list of mandatory attributes and functions  
 9854 and a list of optional features.

9855 **7.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCB 1811 1812 1821
2	CCB 2430

9856 **7.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DRLK	Type 2 (server to client)

9857 **7.3.1.3 Cluster Identifiers**

Identifier	Name
0x0101	Door Lock

9858 **7.3.2 Server**

9859 Generally the door lock itself implements the server side of this cluster. The attributes and commands listed in this  
 9860 cluster were developed to be implemented by a door lock which has the ability to keep track of multiple users and  
 9861 schedules.

9862 **7.3.2.1 Alarms, Reports, and Events**

9863 A door lock implementing all of the optional features provided in this cluster has the ability to push data to a controller  
 9864 in three different forms, Alarms, Reports and Events. Alarms are used to report critical states on the door lock. Reports  
 9865 are used to inform a subscribed device of changes of state in specific attributes on the lock. Events are used to inform  
 9866 a bound device about changes in state related to the operation and programming of the door lock. Event commands  
 9867 are sent to a binding. Examples of events are locking and unlocking the lock and adding or deleting a user on the lock.

### 9868 7.3.2.1.1 Alarms

9869 The door lock cluster provides several alarms which can be sent when there is a critical state on the door lock. The  
9870 alarms available for the door lock cluster are listed in the section below outlining the alarm mask attribute. The Alarm  
9871 cluster is used to generate the actual alarms.

9872 **Alarm example:** If the first bit of the attribute Alarm Mask is set, any device that is bound to the alarm cluster will  
9873 be informed each time the deadbolt becomes jammed. If for some reason the door lock became jammed, the door lock  
9874 would send an alarm command from the alarms cluster with the payload illustrated in Figure 7-2.

9875 **Figure 7-2. Format of the Alarm Cluster**

Octets	1	2
Data Type	enum8	Cluster ID
Field Name	Alarm Code	Cluster Identifier
Field Value	0x00	0x0101

### 9876 7.3.2.1.2 Reports

9877 The reporting mechanism within the ZCL can be used to subscribe to changes in a specific attribute within the door  
9878 lock.

9879 **Report example:** If an application wants to know each time a programming change is made on the door lock, it can  
9880 use the reporting mechanism to be informed of changes to the Operating Mode attribute. Each time the Operating  
9881 Mode changes to programming mode, the application will be informed and can then sync its knowledge of user data  
9882 to make sure that it has an up to date record of user the users supported on the door lock.

### 9883 7.3.2.1.3 Events

9884 The event mechanism described within this document is unique to the Door Lock Cluster and was designed specifically  
9885 for this cluster and no other. It is in part modeled on similar mechanisms in other clusters such as the load control  
9886 events in the Demand Response and Load Control cluster in Smart Energy (DRLC).

9887 The event mechanism in the door lock centers on the transmission of two commands autonomously generated by the  
9888 server and sent to a bound device. The assumption is that the binding mechanism will be used to commission the  
9889 server to send these commands.

9890 There are two types of events on the door lock, operational and programmatic. Operational events relate to the general  
9891 operation of the door lock, when it locks and unlocks for instance. The programmatic events relate to the programming  
9892 of the door lock, for example when users are added or modified via the keypad.

9893 Events are transmitted using two server commands, the Operation Event Notification Command and Programming  
9894 Event Notification Command.

9895 A primary key uniquely identifies each event. The key consists of the event's type (operation, programming etc...),  
9896 source (keypad, RF, manual, etc...) and event code. The event mask bit that matches its type, source and event code  
9897 controls the generation of each event. A complete list of events is included in the description of their commands along  
9898 with the specific attribute and bit that control their generation.

## 9899 7.3.2.2 Door Lock Security

9900 *The following functionality has not been validated at a Specification Validation Event and is therefore considered*  
9901 *provisional.*

9902 Door locks have the ability to require the use of APS encryption for sending and receiving of all cluster messages.  
9903 The Security Level attribute is used to specify the type of encryption required by the door lock.

9904 The APS key MUST be unique to the door lock device to provide the enhanced security needed. Therefore, if APS  
9905 security setting is selected, the device SHALL use a randomly generated install code to generate the unique APS link  
9906 key to join to the network and use this unique APS link key to encryption all Door Lock Cluster, Group Cluster, Scene  
9907 Cluster messages.

9908 The hashing method used to convert install code into APS link key is AES-MMO.

9909 It SHOULD be noted that for the device with unique APS link key to join successfully to the network, the Trust Center  
9910 will need to have a method for the user/installer to input the unique install code for the device.

9911 Note that the security setting will only take effect when the device is not part of a network. If the user modifies the  
9912 Security Level setting while the device is part of a network, the setting will not be applied until the device leaves the  
9913 network and commissions to a network again.

### 9914 **7.3.2.3 Time**

9915 There are various references to the LocalTime within this cluster specification.

9916 LocalTime (32-bit unsigned integer) represents the number of seconds since January 1 2000, in the local zone with  
9917 time saving adjusted.

### 9918 **7.3.2.4 PIN/RFID Code Format**

9919 The PIN/RFID codes defined in this specification are all in ZCL OCTET STRING format. The first octet in the string  
9920 specifies the number of octets contained in the remaining of the data field not including itself.

9921 All value in the PIN/RFID code SHALL be ASCII encoded regardless if the PIN/RFID codes are number or characters.  
9922 For example, code of “1, 2, 3, 4” SHALL be represented as 0x31, 0x32, 0x33, 0x34.

### 9923 **7.3.2.5 Process for Creating a New User with Schedule**

9924 The following is the process that the client device SHALL follow for creating a new user with weekday schedule or  
9925 yearday schedule. The following process SHOULD be implemented as an atomic set and SHOULD not be broken up.

- 9926 1. Set Pin Code
- 9927 2. Set Weekday Schedule or Set Yearday Schedule
- 9928 3. Set User Type to the desired schedule user type.

### 9929 **7.3.2.6 Process for Clearing All Schedules for a User**

9930 The following is the process that the client device SHALL follow for clearing all weekday schedule or all yearday  
9931 schedule for a user. The following process SHOULD be implemented as an atomic set and SHOULD not be broken  
9932 up.

- 9933 1. Clear All Weekday Schedule or Clear All Yearday Schedule
- 9934 2. Set User Type to the Unrestricted User Type

9935 **Note:** If the User Type is not reset to Unrestricted User, the associated user Code (ex: PIN/RFID) will not have access.

### 9936 **7.3.2.7 Clarification of Changing the User Type**

9937 When the user type is changed from a scheduled user to some other user type, the door lock server MAY remove the  
9938 user's schedule.

9939 **7.3.2.8 Clarification for Changing the User Code**

9940 When changing the user code, the server SHALL not require that the user code be removed first.

9941 **7.3.2.9 Server Attributes**

9942 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 9943 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 9944 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 9945 for Door Lock Cluster Server are listed in Table 7-7.

9946 **Table 7-7. Attribute Sets Description**

Attribute Set	Identifier Description
0x0000 – 0x000F	Basic Information Attribute Set
0x0010 – 0x001F	User, PIN, Schedule Information Attribute Set
0x0020 – 0x002F	Operational Settings Attribute Set
0x0030 – 0x003F	Security Settings Attribute Set
0x0040 – 0x004F	Alarm and Event Masks Attribute Set

9947 **7.3.2.10 Basic Information Attribute Set**9948 **Table 7-8. Current Information Attribute Set**

Identifier	Name	Type	Access	Def	M/O
0x0000	<i>LockState</i>	enum8	RP	-	M
0x0001	<i>LockType</i>	enum8	R	-	M
0x0002	<i>ActuatorEnabled</i>	bool	R	-	M
0x0003	<i>DoorState</i>	enum8	RP	-	O
0x0004	<i>DoorOpenEvents</i>	uint32	RW	-	O
0x0005	<i>DoorClosedEvents</i>	uint32	RW	-	O
0x0006	<i>OpenPeriod</i>	uint16	RW	-	O

9949 **7.3.2.10.1 LockState Attribute**

9950 This attribute has the following possible values:

9951 **Table 7-9. LockState Attribute Values**

Value	Definition
0x00	Not fully locked



Value	Definition
0x01	Locked
0x02	Unlocked
0xFF	Undefined

9952

9953 **7.3.2.10.2 LockType Attribute**

9954 The *LockType* attribute is indicated by an enumeration:

9955 **Table 7-10. LockType Attribute Values**

Value	Definition
0x00	Dead bolt
0x01	Magnetic
0x02	Other
0x03	Mortise
0x04	Rim
0x05	Latch Bolt
0x06	Cylindrical Lock
0x07	Tubular Lock
0x08	Interconnected Lock
0x09	Dead Latch
0x0A	Door Furniture

9956 **7.3.2.10.3 ActuatorEnabled Attribute**

9957 This attribute has the following possible values:

9958 **Table 7-11. ActuatorEnabled Attribute Values**

Boolean Value	Definition
0	Disabled
1	Enabled

9959 **7.3.2.10.4 DoorState Attribute**

9960 This attribute has the following possible values:

9961 **Table 7-12. DoorState Attribute Values**

Value	Definition
0x00	Open
0x01	Closed
0x02	Error (jammed)
0x03	Error (forced open)
0x04	Error (unspecified)
0xFF	Undefined

9962 **7.3.2.10.5 DoorOpenEvents Attribute**

9963 This attribute holds the number of door open events that have occurred since it was last zeroed.

9964 **7.3.2.10.6 DoorClosedEvents Attribute**

9965 This attribute holds the number of door closed events that have occurred since it was last zeroed.

9966 **7.3.2.10.7 OpenPeriod Attribute**9967 This attribute holds the number of minutes the door has been open since the last time it transitioned from closed to open.  
99689969 **7.3.2.11 User, PIN, Schedule, Log Information Attribute Set**9970 **Table 7-13. User, PIN, Schedule, Log Information Attribute Set**

Id	Description	Type	Access	Def	M/O
0x0010	<i>NumberOfLogRecordsSupported</i>	uint16	R	0	O
0x0011	<i>NumberOfTotalUsersSupported</i>	uint16	R	0	O
0x0012	<i>NumberOfPINUsersSupported</i>	uint16	R	0	O
0x0013	<i>NumberOfRFIDUsersSupported</i>	uint16	R	0	O
0x0014	<i>NumberOfWeekDaySchedulesSupportedPerUser</i>	uint8	R	0	O
0x0015	<i>NumberOfYearDaySchedulesSupportedPerUser</i>	uint8	R	0	O
0x0016	<i>NumberOfHolidaySchedulesSupported</i>	uint8	R	0	O

<b>Id</b>	<b>Description</b>	<b>Type</b>	<b>Access</b>	<b>Def</b>	<b>M/O</b>
0x0017	<i>MaxPINCodeLength</i>	uint8	R	0x08	O
0x0018	<i>MinPINCodeLength</i>	uint8	R	0x04	O
0x0019	<i>MaxRFIDCodeLength</i>	uint8	R	0x14	O
0x001A	<i>MinRFIDCodeLength</i>	uint8	R	0x08	O

9971 **7.3.2.11.1 NumberOfLogRecordsSupported Attribute**

9972 The number of available log records.

9973 **7.3.2.11.2 NumberOfTotalUsersSupported Attribute**

9974 Number of total users supported by the lock. This value is equal to the higher one of [# of PIN Users Supported] and  
 9975 [# of RFID Users Supported]

9976 **7.3.2.11.3 NumberOfPINUsersSupported Attribute**

9977 The number of PIN users supported.

9978 **7.3.2.11.4 NumberOfRFIDUsersSupported Attribute**

9979 The number of RFID users supported.

9980 **7.3.2.11.5 NumberOfWeekDaySchedulesSupportedPerUser Attribute**

9981 The number of configurable week day schedule supported per user.

9982 **7.3.2.11.6 NumberOfYearDaySchedulesSupportedPerUser Attribute**

9983 The number of configurable year day schedule supported per user.

9984 **7.3.2.11.7 NumberOfHolidaySchedulesSupported Attribute**

9985 The number of holiday schedules supported for the entire door lock device.

9986 **7.3.2.11.8 MaxPINCodeLength Attribute**

9987 An 8 bit value indicates the maximum length in bytes of a PIN Code on this device. The default is set to 8 since most  
 9988 lock manufacturers currently allow PIN Codes of 8 bytes or less.

9989 **7.3.2.11.9 MinPINCodeLength Attribute**

9990 An 8 bit value indicates the minimum length in bytes of a PIN Code on this device. The default is set to 4 since most  
 9991 lock manufacturers do not support PIN Codes that are shorter than 4 bytes.

9992 **7.3.2.11.10 MaxRFIDCodeLength Attribute**

9993 An 8 bit value indicates the maximum length in bytes of a RFID Code on this device. The value depends on the RFID  
9994 code range specified by the manufacturer, if media anti-collision identifiers (UID) are used as RFID code, a value of  
9995 20 (equals 10 Byte ISO 14443A UID) is recommended.

9996 **7.3.2.11.11 MinRFIDCodeLength Attribute**

9997 An 8 bit value indicates the minimum length in bytes of a RFID Code on this device. The value depends on the RFID  
9998 code range specified by the manufacturer, if media anti-collision identifiers (UID) are used as RFID code, a value of  
9999 8 (equals 4 Byte ISO 14443A UID) is recommended.

10000 **7.3.2.12 Operational Settings Attribute Set**

10001 The attributes within this attribute set affect the physical behavior on the server device. Some of the setting might not  
10002 be applicable to the specific device. When the client sends the write attribute request with values that are not applicable  
10003 to the server device, the server SHALL send back a Write Attribute Response with error status not equal to  
10004 ZCL\_SUCCESS(0x00). It is suggested that it SHOULD respond with an error status of ZCL\_INVALID\_VALUE  
10005 (0x87).

10006 **Table 7-14. Operational Settings Attribute Set**

Id	Description	Type	Access	Def	M/O
0x0020	<i>EnableLogging</i>	bool	R*W P	0	O
0x0021	<i>Language</i>	string (3 bytes)	R*W P	0	O
0x0022	<i>LEDSettings</i>	uint8	R*W P	0	O
0x0023	<i>AutoRelockTime</i>	uint32	R*W P	0	O
0x0024	<i>SoundVolume</i>	uint8	R*W P	0	O
0x0025	<i>OperatingMode</i>	enum8	R*W P	0	O
0x0026	<i>SupportedOperatingModes</i>	map16	R	0x0001	O
0x0027	<i>DefaultConfigurationRegister</i>	map16	RP	0x0000	O
0x0028	<i>EnableLocalProgramming</i>	bool	R*W P	0x01	O
0x0029	<i>EnableOneTouchLocking</i>	bool	RWP	0	O
0x002A	<i>EnableInsideStatusLED</i>	bool	RWP	0	O
0x002B	<i>EnablePrivacyModeButton</i>	bool	RWP	0	O

10007 **7.3.2.12.1 EnableLogging Attribute**

10008 Enable/disable event logging. When event logging is enabled, all event messages are stored on the lock for retrieval.  
10009 Logging events can be but not limited to Tamper Alarm, Lock, Unlock, Autolock, User Code Added, User Code  
10010 Deleted, Schedule Added, and Schedule Deleted. For a full detail of all the possible alarms and events, please refer to  
10011 the full list in the Alarm and Event Masks Attribute Set.

10012 **7.3.2.12.2 Language Attribute**

10013 Modifies the language for the on-screen or audible user interface using three bytes from ISO-639-1. It consists of one  
 10014 byte of length and two bytes for the language code. For example if the language is set to English, the value would be  
 10015 "02 65 6E" for the language code "en".

10016 **7.3.2.12.3 OperatingMode Attribute**

10017 Table 7-15 shows the current operating mode and which interfaces are enabled during each of the operating mode.

10018 **Table 7-15. Operating Modes**

Enum	Operating Mode	Interface (E = Enabled; D = Disabled)		
		Keypad	RF	RFID
0x00	Normal	E	E	E
0x01	Vacation	D	E	E
0x02	Privacy	D	D	D
0x03	No RF Lock/Unlock	E	D	E
0x04	Passage	N/A	N/A	N/A

10019

10020 **Normal Mode:** The lock operates normally. All interfaces are enabled.

10021 **Vacation Mode:** Only RF interaction is enabled. The keypad cannot be operated.

10022 **Privacy Mode:** All external interaction with the door lock is disabled. This is intended so that users presumably inside  
 10023 the property will have control over the entrance. Privacy mode assumes that the lock can only be operated from inside  
 10024 by operating the thumb turn or some other means of ending privacy mode.

10025 **No RF Lock or Unlock:** This mode only disables RF interaction with the lock. It specifically applies to the Lock,  
 10026 Unlock, Toggle, and Unlock with Timeout Commands.

10027 **Passage Mode:** The lock is open or can be open or closed at will without the use of a Keypad or other means of user  
 10028 validation.

10029 **Note:** For modes that disable the RF interface, the door lock SHALL respond to Lock, Unlock, Toggle, and Unlock  
 10030 with Timeout commands with a ZCL response with status FAILURE (0x01) and not take the action requested by those  
 10031 commands. The door lock SHALL NOT disable the radio or otherwise unbind or leave the network. It SHALL still  
 10032 respond to all other commands and requests.

10033 **7.3.2.12.4 SupportedOperatingModes Attribute**

10034 This bitmap contains all operating bits of the Operating Mode Attribute supported by the lock. The value of the  
 10035 enumeration in "Operating Mode" defines the related bit to be set, as shown in Table 7-16. All bits supported by a  
 10036 lock SHALL be set to zero.

10037

**Table 7-16. Bit Values for the *SupportedOperatingModes* Attribute**

Bitmap Number	Description
0	Normal Mode Supported
1	Vacation Mode Supported
2	Privacy Mode Supported
3	No RF Lock or Unlock Mode Supported
4	Passage Mode Supported

10038

**7.3.2.12.5 LEDSettings Attribute**

10039

10040 The settings for the LED support three different modes, shown in Table 7-17:

10041

**Table 7-17. Modes for the *LEDSettings* Attribute**

Attribute Identifier	Definition
0x00	Never use LED for signalization
0x01	Use LED signalization except for access allowed events
0x02	Use LED signalization for all events

**7.3.2.12.6 AutoRelockTime Attribute**

10042

10043 The number of seconds to wait after unlocking a lock before it automatically locks again. 0=disabled. If set, unlock  
10044 operations from any source will be timed. For one time unlock with timeout use the specific command.**7.3.2.12.7 SoundVolume Attribute**

10045

10046 The sound volume on a door lock has three possible settings: silent, low and high volumes, shown in Table 7-18.

10047

**Table 7-18. Settings for the *SoundVolume* Attribute**

Attribute Identifier	Definition
0x00	Silent Mode
0x01	Low Volume
0x02	High Volume

10048 **7.3.2.12.8 DefaultConfigurationRegister Attribute**

10049 This attribute represents the default configurations as they are physically set on the device (example: hardware dip  
 10050 switch setting, etc...) and represents the default setting for some of the attributes within this Operational Setting  
 10051 Attribute Set (for example: LED, Auto Lock, Sound Volume, and Operating Mode attributes), as in Table 7-19.

10052 This is a read-only attribute and is intended to allow clients to determine what changes MAY need to be made without  
 10053 having to query all the included attributes. It MAY be beneficial for the clients to know what the device’s original  
 10054 settings were in the event that the device needs to be restored to factory default settings.

10055 If the Client device would like to query and modify the door lock server’s operating settings, it SHOULD send read  
 10056 and write attribute request to the specific attributes.

10057 For example, the Buzzer bitmap within this attribute is off. It represents the hardware dip switch Buzzer setting  
 10058 (original default setting) is off and the Sound Volume attribute default value is in Silent Mode. However, it is possible  
 10059 that the current Sound Volume is in High Volume. Therefore, if the client wants to query/modify the current Sound  
 10060 Volume setting on the server, the client SHOULD read/write to the Sound Volume attribute.

10061 **Table 7-19. DefaultConfigurationRegister Attribute**

Bitmap Number	Description
0	0 - Enable Local Programming Attribute default value is 0 (disabled) 1 - Enable Local Programming Attribute default value is 1 (enabled)
1	0 - Keypad Interface default access is disabled 1 - Keypad Interface default access is enabled
2	0 - RF Interface default access is disabled 1 - RF Interface default access is enabled
5	0 - Sound Volume attribute default value is 0 (Slight Mode) 1 - Sound Volume attribute default value is equal to something other than 0x00
6	0 - Auto Relock Time attribute default value = 0x00 1 - Auto Relock Time attribute default value is equal to something other than 0x00
7	0 - Led Settings attribute default value = 0x00 1 - Led Settings attribute default value is equal to something other than 0x00

10062 **7.3.2.12.9 EnableLocalProgramming Attribute**

10063 Enable/disable local programming on the door lock. The local programming features includes but not limited to adding  
 10064 new user codes, deleting existing user codes, add new schedule, deleting existing schedule on the local door lock  
 10065 interfaces. If this value is set to 0x01 or TRUE then local programming is enabled on the door lock. If it is set to 0x00  
 10066 or FALSE then local programming is disabled on the door lock. Local programming is enabled by default.

10067 **7.3.2.12.10 EnableOneTouchLocking Attribute**

10068 Enable/disable the ability to lock the door lock with a single touch on the door lock.

10069 **7.3.2.12.11 EnableInsideStatusLED Attribute**

10070 Enable/disable an inside LED that allows the user to see at a glance if the door is locked.

10071 **7.3.2.12.12 EnablePrivacyModeButton Attribute**10072 Enable/disable a button inside the door that is used to put the lock into privacy mode. When the lock is in privacy  
10073 mode it cannot be manipulated from the outside.10074 **7.3.2.13 Security Settings Attribute Set**10075 **Table 7-20. Security Settings Attribute Set**

Id	Description	Type	Access	Def	M/O
0x0030	<i>WrongCodeEntryLimit</i>	uint8	R*W P	0	O
0x0031	<i>UserCodeTemporaryDisableTime</i>	uint8	R*W P	0	O
0x0032	<i>SendPINOverTheAir</i>	bool	R*W P	0	O
0x0033	<i>RequirePINforRFOperation</i>	bool	R*W P	0	O
0x0034	<i>SecurityLevel</i>	enum8	RP	0	O

10076 **7.3.2.13.1 WrongCodeEntryLimit Attribute**10077 The number of incorrect codes or RFID presentment attempts a user is allowed to enter before the door will enter a  
10078 lockout state. The lockout state will be for the duration of *UserCodeTemporaryDisableTime*.10079 **7.3.2.13.2 UserCodeTemporaryDisableTime Attribute**10080 The number of seconds that the lock shuts down following wrong code entry. 1-255 seconds. Device can shut down  
10081 to lock user out for specified amount of time. (Makes it difficult to try and guess a PIN for the device.)10082 **7.3.2.13.3 SendPINOverTheAir Attribute**10083 Boolean set to True if it is ok for the door lock server to send PINs over the air. This attribute determines the behavior  
10084 of the server's TX operation. If it is false, then it is not ok for the device to send PIN in any messages over the air.10085 The PIN field within any door lock cluster message SHALL keep the first octet unchanged and masks the actual code  
10086 by replacing with 0xFF. For example (PIN "1234"): If the attribute value is True, 0x04 0x31 0x32 0x33 0x34 SHALL  
10087 be used in the PIN field in any door lock cluster message payload. If the attribute value is False, 0x04 0xFF 0xFF  
10088 0xFF 0xFF SHALL be used.10089 **7.3.2.13.4 RequirePINForRFOperation Attribute**10090 Boolean set to True if the door lock server requires that an optional PINs be included in the payload of RF lock  
10091 operation events like Lock, Unlock and Toggle in order to function.



10092 **7.3.2.13.5 SecurityLevel Attribute**

10093 Door locks MAY sometimes wish to implement a higher level of security within the application protocol in addition  
 10094 to the default network security. For instance, a door lock MAY wish to use additional APS security for cluster  
 10095 transactions. This protects the door lock against being controlled by any other devices which have access to the  
 10096 network key.

10097 The Security Level attribute allows the door lock manufacturer to indicate what level of security the door lock requires.

10098 There are two levels of security possible within this cluster:

- 10099 • 0 = Network Security (default)
- 10100 • 1 = APS Security

10101 This attribute is read only over the network to protect security method defined by each manufacturer.

10102 However, manufacturer can provide method to modify the security setting locally on the device. The security setting  
 10103 modification will not take effect when the device is in a network.

10104 **7.3.2.14 Alarm and Event Masks Attribute Set**

10105 **Table 7-21. Alarm and Event Masks Attribute Set**

<b>Id</b>	<b>Description</b>	<b>Type</b>	<b>Access</b>	<b>Default</b>	<b>M/O</b>
0x0040	<i>AlarmMask</i>	map16	RWP	0x0000	O
0x0041	<i>KeypadOperationEventMask</i>	map16	RWP	0x0000	O
0x0042	<i>RFOperationEventMask</i>	map16	RWP	0x0000	O
0x0043	<i>ManualOperationEventMask</i>	map16	RWP	0x0000	O
0x0044	<i>RFIDOperationEventMask</i>	map16	RWP	0x0000	O
0x0045	<i>KeypadProgrammingEventMask</i>	map16	RWP	0x0000	O
0x0046	<i>RFProgrammingEventMask</i>	map16	RWP	0x0000	O
0x0047	<i>RFIDProgrammingEventMask</i>	map16	RWP	0x0000	O

10106 **7.3.2.14.1 AlarmMask Attribute**

10107 The alarm mask is used to turn on/off alarms for particular functions, as shown in Table 7-22. Alarms for an alarm  
 10108 group are enabled if the associated alarm mask bit is set. Each bit represents a group of alarms. Entire alarm groups  
 10109 can be turned on or off by setting or clearing the associated bit in the alarm mask.

10110 **Table 7-22. Alarm Code Table**

<b>Alarm Code</b>	<b>Bitmap Number</b>	<b>Alarm Condition</b>
0x00	0	Deadbolt Jammed
0x01	1	Lock Reset to Factory Defaults
0x02	2	Reserved
0x03	3	RF Module Power Cycled

Alarm Code	Bitmap Number	Alarm Condition
0x04	4	Tamper Alarm – wrong code entry limit
0x05	5	Tamper Alarm - front escutcheon removed from main
0x06	6	Forced Door Open under Door Locked Condition

#### 10111 **7.3.2.14.2 KeypadOperationEventMask Attribute**

10112 Event mask used to turn on and off the transmission of keypad operation events. This mask DOES NOT apply to the  
10113 storing of events in the report table.

10114 For detail event mask value, please refer to Table 7-30.

#### 10115 **7.3.2.14.3 RFOperationEventMask Attribute**

10116 Event mask used to turn on and off the transmission of RF operation events. This mask DOES NOT apply to the  
10117 storing of events in the report table.

10118 For detail event mask value, please refer to Table 7-31.

#### 10119 **7.3.2.14.4 ManualOperationEventMask Attribute**

10120 Event mask used to turn on and off manual operation events. This mask DOES NOT apply to the storing of events in  
10121 the report table.

10122 For detail event mask value, please refer to Table 7-32.

#### 10123 **7.3.2.14.5 RFIDOperationEventMask Attribute**

10124 Event mask used to turn on and off RFID operation events. This mask DOES NOT apply to the storing of events in  
10125 the report table.

10126 For detail event mask value, please refer to Table 7-33.

#### 10127 **7.3.2.14.6 KeypadProgrammingEventMask Attribute**

10128 Event mask used to turn on and off keypad programming events. This mask DOES NOT apply to the storing of events  
10129 in the report table.

10130 For detail event mask value, please refer to Table 7-36.

#### 10131 **7.3.2.14.7 RFProgrammingEventMask Attribute**

10132 Event mask used to turn on and off RF programming events. This mask DOES NOT apply to the storing of events in  
10133 the report table.

10134 For detail event mask value, please refer to Table 7-37.

#### 10135 **7.3.2.14.8 RFIDProgrammingEventMask Attribute**

10136 Event mask used to turn on and off RFID programming events. This mask DOES NOT apply to the storing of events  
10137 in the report table.

10138 For detail event mask value, please refer to Table 7-38.

10139 **7.3.2.15 Server Commands Received**

10140 The commands received by the server are listed in Table 7-23.

10141 **Table 7-23. Commands Received by the Server Cluster**

Command ID	Description	M/O
0x00	Lock Door	M
0x01	Unlock Door	M
0x02	Toggle	O
0x03	Unlock with Timeout	O
0x04	Get Log Record	O
0x05	Set PIN Code	O
0x06	Get PIN Code	O
0x07	Clear PIN Code	O
0x08	Clear All PIN Codes	O
0x09	Set User Status	O
0x0A	Get User Status	O
0x0B	Set Weekday Schedule	O
0x0C	Get Weekday Schedule	O
0x0D	Clear Weekday Schedule	O
0x0E	Set Year Day Schedule	O
0x0F	Get Year Day Schedule	O
0x10	Clear Year Day Schedule	O
0x11	Set Holiday Schedule	O
0x12	Get Holiday Schedule	O
0x13	Clear Holiday Schedule	O
0x14	Set User Type	O

Command ID	Description	M/O
0x15	Get User Type	O
0x16	Set RFID Code	O
0x17	Get RFID Code	O
0x18	Clear RFID Code	O
0x19	Clear All RFID Codes	O

### 10142 7.3.2.15.1 Lock Door Command

10143 This command causes the lock device to lock the door. As of HA 1.2, this command includes an optional code for the  
10144 lock. The door lock MAY require a PIN depending on the value of the [Require PIN for RF Operation attribute].

10145 **Figure 7-3. Format of the Lock Door Command**

Octets	Variable
Data Type	octstr
Field Name	PIN/RFID Code

### 10146 7.3.2.15.2 Unlock Door Command

10147 This command causes the lock device to unlock the door. As of HA 1.2, this command includes an optional code for the  
10148 lock. The door lock MAY require a code depending on the value of the [Require PIN for RF Operation attribute].

10149 **Note:** If the attribute *AutoRelockTime* is supported the lock will close when the auto relock time has expired.

10150 **Figure 7-4. Format of the Unlock Door Command**

Octets	Variable
Data Type	octstr
Field Name	PIN/RFID Code

### 10151 7.3.2.15.3 Toggle Command

10152 Request the status of the lock. As of HA 1.2, this command includes an optional code for the lock. The door lock  
10153 MAY require a code depending on the value of the [Require PIN for RF Operation attribute].

10154 **Figure 7-5. Format of the Toggle Command**

Octets	Variable
Data Type	octstr

<b>Field Name</b>	PIN/RFID Code
-------------------	---------------

10155 **7.3.2.15.4 Unlock with Timeout Command**

10156 This command causes the lock device to unlock the door with a timeout parameter. After the time in seconds specified  
 10157 in the timeout field, the lock device will relock itself automatically. This timeout parameter is only temporary for this  
 10158 message transition only and overrides the default relock time as specified in the [Auto Relock Time attribute] attribute.  
 10159 If the door lock device is not capable of or does not want to support temporary Relock Timeout, it SHOULD not  
 10160 support this optional command.

10161 **Figure 7-6. Format of the Unlock with Timeout Command**

<b>Octets</b>	1	Variable
<b>Data Type</b>	uint16	octstr
<b>Field Name</b>	Timeout in seconds	PIN/RFID Code

10162 **7.3.2.15.5 Get Log Record Command**

10163 Request a log record. Log number is between 1 – [Number of Log Records Supported attribute]. If log number 0 is  
 10164 requested then the most recent log entry is returned.

10165 **Figure 7-7. Format of the Get Log Record Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Log Index

10166  
 10167 **Log record format:** The log record format is defined in the description of the Get Log Record Response command.

10168 **7.3.2.15.6 User Status and User Type Values**

10169 The following User Status and User Type values are used in the payload of multiple commands:

10170 **User Status:** Used to indicate what the status is for a specific user ID.

10171 **Table 7-24. User Status Value**

<b>Enum</b>	<b>Description</b>
0	Available
1	Occupied / Enabled
3	Occupied / Disabled
0xff	Not Supported

10172

10173 **User Type:** Used to indicate what the type is for a specific user ID.

10174

**Table 7-25. User Type Value**

Enum	Description
0	Unrestricted User (default)
1	Year Day Schedule User
2	Week Day Schedule User
3	Master User
4	Non Access User
0xff	Not Supported

10175

10176 **Unrestricted:** User has access 24/7 provided proper PIN is supplied (e.g., owner). Unrestricted user type is the default  
10177 user type.10178 **Year Day Schedule User:** User has ability to open lock within a specific time period (e.g., guest).10179 **Week Day Schedule User:** User has ability to open lock based on specific time period within a reoccurring weekly  
10180 schedule (e.g., cleaning worker).10181 **Master User:** User has ability to both program and operate the door lock. This user can manage the users and user  
10182 schedules. In all other respects this user matches the unrestricted (default) user. Master user is the only user that can  
10183 disable the user interface (keypad, RF, etc...).10184 **Non Access User:** User is recognized by the lock but does not have the ability to open the lock. This user will only  
10185 cause the lock to generate the appropriate event notification to any bound devices.10186 **7.3.2.15.7 Set PIN Code Command**

10187 Set a PIN into the lock.

10188

**Figure 7-8. Format of the Set PIN Code Command**

<b>Octets</b>	2	1	1	Variable
<b>Data Type</b>	uint16	uint8	enum8	octstr
<b>Field Name</b>	User ID	User Status	User Type	PIN

10189

10190 User ID is between 0 - [# of PIN Users Supported attribute]. Only the values 1 (Occupied/Enabled) and 3  
10191 (Occupied/Disabled) are allowed for User Status.10192 **7.3.2.15.8 Get PIN Code Command**

10193 Retrieve a PIN Code. User ID is between 0 - [# of PIN Users Supported attribute].

10194

**Figure 7-9. Format of the Get PIN Code Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	User ID

10195 **7.3.2.15.9 Clear PIN Code Command**

10196 Delete a PIN. User ID is between 0 - [# of PIN Users Supported attribute].

10197

**Figure 7-10. Format of the Clear PIN Code Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	User ID

10198 Note: If you delete a PIN Code and this user didn't have a RFID Code, the user status is set to "0 Available", the user  
 10199 type is set to the default value and all schedules are also set to the default values.

10200 **7.3.2.15.10 Clear All PIN Codes Command**

10201 Clear out all PINs on the lock.

10202 **Note:** *On the server, the clear all PIN codes command SHOULD have the same effect as the Clear PIN Code command*  
 10203 *with respect to the setting of user status, user type and schedules.*

10204 **7.3.2.15.11 Set User Status Command**

10205 Set the status of a user ID. User Status value of 0x00 is not allowed. In order to clear a user id, the Clear ID Command  
 10206 SHALL be used. For user status value please refer to User Status Value.

10207

**Figure 7-11. Format of the Set User Status Command**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	User ID	User Status

10208 **7.3.2.15.12 Get User Status Command**

10209 Get the status of a user.

10210

**Figure 7-12. Format of the Get User Status Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	User ID

10211 **7.3.2.15.13 Set Week Day Schedule Command**

10212 Set a weekly repeating schedule for a specified user.

10213

**Figure 7-13. Format of the Set Week Day Schedule Command**

<b>Octets</b>	1	2	1	1	1	1	1
<b>Data Type</b>	uint8	uint16	map8	uint8	uint8	uint8	uint8
<b>Field Name</b>	Schedule ID #	User ID	Days Mask	Start Hour	Start Minute	End Hour	End Minute

10214

10215 **Schedule ID:** number is between 0 – [# of Week Day Schedules Per User attribute].10216 **User ID:** is between 0 – [# of Total Users Supported attribute].10217 **Days Mask:** bitmask of the effective days in the order XSFTWTMS.

10218

**Figure 7-14. Format of Days Mask Bits**

<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
Reserved	Sat	Fri	Thur	Wed	Tue	Mon	Sun

10219 Days mask is listed as bitmask for flexibility to set same schedule across multiple days. For the door lock that does  
 10220 not support setting schedule across multiple days within one command, it SHOULD respond with ZCL  
 10221 INVALID\_FIELD (0x85) status when received the set schedule command days bitmask field has multiple days  
 10222 selected.

10223 **Start Hour:** in decimal format represented by 0x00 – 0x17 (00 to 23 hours).10224 **Start Minute:** in decimal format represented by 0x00 – 0x3B (00 to 59 mins).

10225 **End Hour:** in decimal format represented by 0x00 – 0x17 (00 to 23 hours). End Hour SHALL be equal or greater  
 10226 than Start Hour.

10227 **End Minute:** in decimal format represented by 0x00 – 0x3B (00 to 59 mins).

10228 If End Hour is equal with Start Hour, End Minute SHALL be greater than Start Minute.

10229 When the Server Device receives the command, the Server Device MAY change the user type to the specific schedule  
 10230 user type. Please refer to section 7.3.2.5, Process for Creating a New User with Schedule, at the beginning of this  
 10231 cluster.



10232 **7.3.2.15.14 Get Week Day Schedule Command**

10233 Retrieve the specific weekly schedule for the specific user.

10234 **Figure 7-15. Format of the Get Week Day Schedule Command**

<b>Octets</b>	1	2
<b>Data Type</b>	uint8	uint16
<b>Field Name</b>	Schedule ID	User ID

10235 **7.3.2.15.15 Clear Week Day Schedule Command**

10236 Clear the specific weekly schedule for the specific user.

10237 **Figure 7-16. Format of the Clear Week Day Schedule Command**

<b>Octets</b>	1	2
<b>Data Type</b>	uint8	uint16
<b>Field Name</b>	Schedule ID	User ID

10238 **7.3.2.15.16 Set Year Day Schedule Command**

10239 Set a time-specific schedule ID for a specified user.

10240 **Figure 7-17. Format of the Set Year Day Schedule Command**

<b>Octets</b>	1	2	4	4
<b>Data Type</b>	uint8	uint16	uint32	uint32
<b>Field Name</b>	Schedule ID	User ID	Local Start Time	Local End Time

10241

10242 Schedule ID number is between 0 – [# of Year Day Schedules Supported Per User attribute]. User ID is between 0 –

10243 [# of Total Users Supported attribute].

10244 Start time and end time are given in LocalTime. End time must be greater than the start time.

10245 When the Server Device receives the command, the Server Device MAY change the user type to the specific schedule

10246 user type. Please refer to Process for Creating a New User with Schedule at the beginning of this cluster.

10247 **7.3.2.15.17 Get Year Day Schedule Command**

10248 Retrieve the specific year day schedule for the specific user.

10249

**Figure 7-18. Format of the Get Year Day Schedule Command**

<b>Octets</b>	1	2
<b>Data Type</b>	uint8	uint16
<b>Field Name</b>	Schedule ID	User ID

### 10250 **7.3.2.15.18 Clear Year Day Schedule Command**

10251 Clears the specific year day schedule for the specific user.

10252

**Figure 7-19. Format of the Clear Year Day Schedule Command**

<b>Octets</b>	1	2
<b>Data Type</b>	uint8	uint16
<b>Field Name</b>	Schedule ID	User ID

### 10253 **7.3.2.15.19 Set Holiday Schedule Command**

10254 Set the holiday Schedule by specifying local start time and local end time with respect to any Lock Operating Mode.

10255

**Figure 7-20. Format of the Set Holiday Schedule Command**

<b>Octets</b>	1	4	4	1
<b>Data Type</b>	uint8	uint32	uint32	enum8
<b>Field Name</b>	Holiday Schedule ID	Local Start Time	Local End Time	Operating Mode During Holiday

10256

10257 Holiday Schedule ID number is between 0 – [# of Holiday Schedules Supported attribute].

10258 Start time and end time are given in LocalTime. End time must be greater than the start time.

10259 Operating Mode is valid enumeration value as listed in operating mode attribute.

### 10260 **7.3.2.15.20 Get Holiday Schedule Command**

10261 Get the holiday Schedule by specifying Holiday ID.

10262

**Figure 7-21. Format of the Get Holiday Schedule Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Holiday Schedule ID

10263 **7.3.2.15.21 Clear Holiday Schedule Command**

10264 Clear the holiday Schedule by specifying Holiday ID.

10265

**Figure 7-22. Format of the Clear Holiday Schedule Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Holiday Schedule ID

10266 **7.3.2.15.22 Set User Type Command**

10267 Set the type byte for a specified user.

10268 For user type value please refer to User Type Value.

10269

**Figure 7-23. Format of the Set User Type Command**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	enum8
<b>Field Name</b>	User ID	User Type

10270 **7.3.2.15.23 Get User Type Command**

10271 Retrieve the type byte for a specific user.

10272

**Figure 7-24. Format of the Get User Type Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	User ID

10273 **7.3.2.15.24 Set RFID Code Command**

10274 Set an ID for RFID access into the lock.

10275

**Figure 7-25. Format of the Set RFID Code Command**

Octets	2	1	1	Variable
Data Type	uint16	uint8	enum8	octstr
Field Name	User ID	User Status	User Type	RFID Code

10276

10277 **User ID:** Between 0 - [# of RFID Users Supported attribute]. Only the values 1 (Occupied/Enabled) and 3  
 10278 (Occupied/Disabled) are allowed for User Status.

10279 **User Status:** Used to indicate what the status is for a specific user ID. The values are according to “Set PIN” while  
 10280 not all are supported.

10281

**Table 7-26. User Status Byte Values for Set RFID Code Command**

User Status Byte	Value
Occupied / Enabled (Access Given)	1
Occupied / Disabled	3
Not Supported	0xff

10282

10283 **User Type:** The values are the same as used for “Set PIN Code.”

### 10284 **7.3.2.15.25 Get RFID Code Command**

10285 Retrieve an ID. User ID is between 0 - [# of RFID Users Supported attribute].

10286

**Figure 7-26. Format of the Get RFID Code Command**

Octets	2
Data Type	uint16
Field Name	User ID

### 10287 **7.3.2.15.26 Clear RFID Code Command**

10288 Delete an ID. User ID is between 0 - [# of RFID Users Supported attribute]. If you delete a RFID code and this user  
 10289 didn't have a PIN code, the user status has to be set to "0 Available", the user type has to be set to the default value,  
 10290 and all schedules which are supported have to be set to the default values.

10291

**Figure 7-27. Format of the Clear RFID Code Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	User ID

10292 **7.3.2.15.27 Clear All RFID Codes Command**

10293 Clear out all RFIDs on the lock. If you delete all RFID codes and this user didn't have a PIN code, the user status has  
 10294 to be set to "0 Available", the user type has to be set to the default value, and all schedules which are supported have  
 10295 to be set to the default values.

10296 **7.3.2.16 Server Commands Generated**

10297 The commands generated by the server are listed in Table 7-27.

10298 **Table 7-27. Commands Generated by the Server Cluster**

<b>Command ID</b>	<b>Description</b>	<b>M/O</b>
0x00	Lock Door Response	M
0x01	Unlock Door Response	M
0x02	Toggle Response	O
0x03	Unlock with Timeout Response	O
0x04	Get Log Record Response	O
0x05	Set PIN Code Response	O
0x06	Get PIN Code Response	O
0x07	Clear PIN Code Response	O
0x08	Clear All PIN Codes Response	O
0x09	Set User Status Response	O
0x0A	Get User Status Response	O
0x0B	Set Weekday Schedule Response	O
0x0C	Get Weekday Schedule Response	O
0x0D	Clear Weekday Schedule Response	O

Command ID	Description	M/O
0x0E	Set Year Day Schedule Response	O
0x0F	Get Year Day Schedule Response	O
0x10	Clear Year Day Schedule Response	O
0x11	Set Holiday Schedule Response	O
0x12	Get Holiday Schedule Response	O
0x13	Clear Holiday Schedule Response	O
0x14	Set User Type Response	O
0x15	Get User Type Response	O
0x16	Set RFID Code Response	O
0x17	Get RFID Code Response	O
0x18	Clear RFID Code Response	O
0x19	Clear All RFID Codes Response	O
0x20	Operating Event Notification	O
0x21	Programming Event Notification	O

### 10299 7.3.2.16.1 Lock Door Response Command

10300 This command is sent in response to a Lock command with one status byte payload. The Status field SHALL be set  
10301 to SUCCESS or FAILURE (see 2.5).

10302 The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the  
10303 client SHOULD query to [Lock State attribute] and [Door State attribute].

10304

**Figure 7-28. Format of the Lock Door Response Command Payload**

<b>Bits</b>	8
<b>Data Type</b>	enum8
<b>Field Name</b>	Status

10305 **7.3.2.16.2 Unlock Door Response Command**

10306 This command is sent in response to a Toggle command with one status byte payload. The Status field SHALL be set  
 10307 to SUCCESS or FAILURE (see 2.5).

10308 The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the  
 10309 client SHOULD query to [Lock State attribute] and [Door State attribute].

10310

**Figure 7-29. Format of the Unlock Door Response Command Payload**

<b>Bits</b>	8
<b>Data Type</b>	enum8
<b>Field Name</b>	Status

10311 **7.3.2.16.3 Toggle Response Command**

10312 This command is sent in response to a Toggle command with one status byte payload. The Status field SHALL be set  
 10313 to SUCCESS or FAILURE (see 2.5).

10314 The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the  
 10315 client SHOULD query to [Lock State attribute] and [Door State attribute].

10316 **7.3.2.16.4 Unlock with Timeout Response Command**

10317 This command is sent in response to an Unlock with Timeout command with one status byte payload. The Status field  
 10318 SHALL be set to SUCCESS or FAILURE (see 2.5).

10319 The status byte only indicates if the message has received successfully. To determine the lock and/or door status, the  
 10320 client SHOULD query to [Lock State attribute] and [Door State attribute].

10321 **7.3.2.16.5 Get Log Record Response Command**

10322 Returns the specified log record. If an invalid log entry ID was requested, it is set to 0 and the most recent log entry  
 10323 will be returned.

10324

**Figure 7-30. Format of the Get Log Record Response Command**

Octets	2	4	1	1	1	2	Variable
Data Type	uint16	uint32	enum8	uint8	uint8	uint16	octstr <sup>113</sup>
Field Name	Log Entry ID	Timestamp	Event Type	Source (see Operation Event Sources)	Event ID/Alarm Code (see Operation Event Codes)	User ID	PIN

10325

10326 **Log Entry ID:** the index into the log table where this log entry is stored. If the log entry requested is 0, the most recent  
10327 log is returned with the appropriate log entry ID.

10328 **Timestamp:** A LocalTime used to timestamp all events and alarms on the door lock.

10329 **Event Type:** Indicates the type of event that took place on the door lock.

10330 0x00 = Operation

10331 0x01 = Programming

10332 0x02 = Alarm

10333 **Source:** A source value where available sources are:

10334 0x00 = Keypad

10335 0x01 = RF

10336 0x02 = Manual

10337 0x03 = RFID

10338 0xff = Indeterminate

10339 If the Event type is 0x02 (Alarm) then the source SHOULD be but does not have to be 0xff (Indeterminate).

10340 **Event ID:** A one byte value indicating the type of event that took place on the door lock depending on the event code  
10341 table provided for a given event type and source.

10342 **User ID:** A two byte value indicating the ID of the user who generated the event on the door lock if one is available.  
10343 If none is available, 0xffff has to be used.

10344 **PIN / ID:** A string indicating the PIN code or RFID code that was used to create the event on the door lock if one is  
10345 available.

### 10346 **7.3.2.16.6 Set PIN Code Response Command**

10347 Returns status of the PIN set command. Possible values are:

10348 0 = Success

10349 1 = General failure

10350 2 = Memory full

10351 3 = Duplicate Code error

<sup>113</sup> CCB 2430 PIN/RFID type is Octet String (octstr)



10352

**Figure 7-31. Format of the Set PIN Code Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status

10353 **7.3.2.16.7 Get PIN Code Response Command**

10354 Returns the PIN for the specified user ID.

10355

**Figure 7-32. Format of the Get PIN Code Response Command**

<b>Octets</b>	2	1	1	Variable
<b>Data Type</b>	uint16	uint8	enum8	octstr
<b>Field Name</b>	User ID	User Status	User Type	Code

10356 If the requested UserId is valid and the Code doesn't exist, Get RFID Code Response SHALL have the following  
 10357 format:

10358 UserId = requested UserId

10359 UserStatus = 0 (available)

10360 UserType = 0xFF (not supported)

10361 RFID = 0 (zero length)

10362 If the requested UserId is invalid, send Default Response with an error status not equal to ZCL\_SUCCESS(0x00).

10363 **7.3.2.16.8 Clear PIN Code Response Command**

10364 Returns pass/fail of the command.

10365

**Figure 7-33. Format of the Clear PIN Code Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10366 **7.3.2.16.9 Clear All PIN Codes Response Command**

10367 Returns pass/fail of the command.

10368

**Figure 7-34. Format of the Clear All PIN Codes Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

**10369 7.3.2.16.10 Set User Status Response Command**

10370 Returns the pass or fail value for the setting of the user status.

10371

**Figure 7-35. Format of the Set User Status Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

**10372 7.3.2.16.11 Get User Status Response Command**

10373 Returns the user status for the specified user ID.

10374

**Figure 7-36. Format of the Get User Status Response Command**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	User ID	User Status

**10375 7.3.2.16.12 Set Week Day Schedule Response Command**

10376 Returns pass/fail of the command.

10377

**Figure 7-37. Format of the Set Week Day Schedule Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10378 **7.3.2.16.13 Get Week Day Schedule Response Command**

10379 Returns the weekly repeating schedule data for the specified schedule ID.

10380 **Figure 7-38. Format of the Get Week Day Schedule Response Command**

<b>Octets</b>	1	2	1	0/1	0/1	0/1	0/1	0/1
<b>Data Type</b>	uint8	uint16	uint8	uint8	uint8	uint8	uint8	uint8
<b>Field Name</b>	Schedule ID	User ID	Status	Days Mask	Start Hour	Start Minute	End Hour	End Minute

10381 **7.3.2.16.13.1 Schedule ID Field**

10382 The requested Schedule ID.

10383 **7.3.2.16.13.2 User ID Field**

10384 The requested User ID.

10385 **7.3.2.16.13.3 Status**

10386 ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

10387 ZCL INVALID\_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

10388 ZCL NOT\_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not  
 10389 corresponding schedule entry found.

10390 Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the  
 10391 fields up to the status field SHALL be present.

10392 **7.3.2.16.13.4 Days Mask**

10393 Days mask is a bitmask of the effective days in the order [E]SMT WTFS. Bit 7 indicates the enabled status of the  
 10394 schedule ID, with the lower 7 bits indicating the effective days mask.

10395 **Figure 7-39. Format of Days Mask Bits**

<b>7</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
EN	Sun	Mon	Tue	Wed	Thu	Fri	Sat

10396 Bit 7: Enabled status: 1=enabled, 0=disabled

10397 **7.3.2.16.13.5 Start Hour**

10398 The Start Hour of the Week Day Schedule: 0-23

10399 **7.3.2.16.13.6 Start Minute**

10400 The Start Min of the Week Day Schedule: 0-59

10401 **7.3.2.16.13.7 End Hour**

10402 The End Hour of the Week Day Schedule: 0-23, must be greater than Start Hour

10403 **7.3.2.16.13.8 End Minute**

10404 The End Min of the Week Day Schedule: 0-59

10405 **7.3.2.16.14 Clear Week Day Schedule ID Response Command**

10406 Returns pass/fail of the command.

10407 **Figure 7-40. Format of the Clear Week Day Schedule ID Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10408 **7.3.2.16.15 Set Year Day Schedule Response Command**

10409 Returns pass/fail of the command.

10410 **Figure 7-41. Format of the Set Year Day Schedule Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10411 **7.3.2.16.16 Get Year Day Schedule Response Command**

10412 Returns the weekly repeating schedule data for the specified schedule ID.

10413

**Figure 7-42. Format of the Get Year Day Schedule Response Command**

<b>Octets</b>	1	2	1	0/4	0/4
<b>Data Type</b>	uint8	uint16	uint8	uint32	uint32
<b>Field Name</b>	Schedule ID	User ID	Status	Local Start Time	Local End Time

10414 **7.3.2.16.16.1 Schedule ID Field**

10415 The requested Schedule ID.

10416 **7.3.2.16.16.2 User ID Field**

10417 The requested User ID.

10418 **7.3.2.16.16.3 Status**

10419 ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

10420 ZCL INVALID\_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

10421 ZCL NOT\_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not  
 10422 corresponding schedule entry found.

10423 Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the  
 10424 fields up to the status field SHALL be present.

10425 **7.3.2.16.16.4 Local Start Time**

10426 Start Time of the Year Day Schedule representing by LocalTime.

10427 **7.3.2.16.16.5 Local End Time**

10428 End Time of the Year Day Schedule representing by LocalTime.

10429 **7.3.2.16.17 Clear Year Day Schedule Response Command**

10430 Returns pass/fail of the command.

10431

**Figure 7-43. Format of the Clear Year Day Schedule Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10432 **7.3.2.16.18 Set Holiday Schedule Response Command**

10433 Returns pass/fail of the command.

10434

**Figure 7-44. Format of the Set Holiday Schedule Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

**10435 7.3.2.16.19 Get Holiday Schedule Response Command**

10436 Returns the Holiday Schedule Entry for the specified Holiday ID.

10437

**Figure 7-45. Format of the Get Holiday Schedule Response Command**

<b>Octets</b>	1	1	0/4	0/4	0/1
<b>Data Type</b>	uint8	uint8	uint32	uint32	enum8
<b>Field Name</b>	Holiday Schedule ID	Status	Local Start Time	Local End Time	Operating Mode During Holiday

**10438 7.3.2.16.19.1 Holiday Schedule ID**

10439 The requested Holiday Schedule ID

**10440 7.3.2.16.19.2 Status**

10441 ZCL SUCCESS (0x00) if both Schedule ID and User ID are valid and there is a corresponding schedule entry.

10442 ZCL INVALID\_FIELD (0x85) if either Schedule ID and/or User ID values are not within valid range

10443 ZCL NOT\_FOUND (0x8B) if both Schedule ID and User ID are within the valid range, however, there is not  
10444 corresponding schedule entry found.10445 Only if the status is ZCL SUCCESS that other remaining fields are included. For other (error) status values, only the  
10446 fields up to the status field SHALL be present.**10447 7.3.2.16.19.3 Local Start Time**

10448 Start Time of the Year Day Schedule representing by LocalTime.

**10449 7.3.2.16.19.4 Local End Time**

10450 End Time of the Year Day Schedule representing by LocalTime.

**10451 7.3.2.16.19.5 Operating Mode**

10452 Operating Mode is valid enumeration value as listed in operating mode attribute.

**10453 7.3.2.16.20 Clear Holiday Schedule Response Command**

10454 Returns pass/fail of the command.

10455

**Figure 7-46. Format of the Clear Holiday Schedule Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10456 **7.3.2.16.21 Set User Type Response Command**

10457 Returns the pass or fail value for the setting of the user type.

10458

**Figure 7-47. Format of the Set User Type Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10459 **7.3.2.16.22 Get User Type Response Command**

10460 Returns the user type for the specified user ID.

10461

**Figure 7-48. Format of the Get User Type Response Command**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	enum8
<b>Field Name</b>	User ID	User Type

10462 **7.3.2.16.23 Set RFID Code Response Command**

10463 Returns status of the Set RFID Code command. Possible values are:

10464 0 = Success

10465 1 = General failure

10466 2 = Memory full

10467 3 = Duplicate ID error

10468

**Figure 7-49. Format of the Set RFID Code Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status

**10469 7.3.2.16.24 Get RFID Code Response Command**

10470 Returns the RFID code for the specified user ID.

10471

**Figure 7-50. Format of the Get RFID Code Response Command**

<b>Octets</b>	2	1	1	Variable
<b>Data Type</b>	uint16	uint8	enum8	octstr
<b>Field Name</b>	User ID	User Status	User Type	RFID Code

10472

10473 If the requested UserId is valid and the Code doesn't exist, Get RFID Code Response SHALL have the following  
10474 format:

10475 UserId = requested UserId

10476 UserStatus = 0 (available)

10477 UserType = 0xFF (not supported)

10478 RFID = 0 (zero length)

10479 If requested UserId is invalid, send Default Response with an error status not equal to ZCL\_SUCCESS(0x00).

**10480 7.3.2.16.25 Clear RFID Code Response Command**

10481 Returns pass/fail of the command.

10482

**Figure 7-51. Format of the Clear RFID Code Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

**10483 7.3.2.16.26 Clear All RFID Codes Response Command**

10484 Returns pass/fail of the command.



10485

**Figure 7-52. Format of the Clear All RFID Codes Response Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Status
<b>Field Value</b>	0=pass 1=fail

10486 **7.3.2.16.27 Operation Event Notification Command**

10487 The door lock server sends out operation event notification when the event is triggered by the various event sources.  
 10488 The specific operation event will only be sent out if the associated bitmask is enabled in the various attributes in the  
 10489 Event Masks Attribute Set.

10490 All events are optional.

10491 **Figure 7-53. Format of the Operation Event Notification Command**

<b>Octets</b>	1	1	2	1	4	Variable/0
<b>Data Type</b>	uint8	uint8	uint16	octstr <sup>114</sup>	uint32	string
<b>Field Name</b>	Operation Event Source	Operation Event Code	User ID	PIN	LocalTime	Data

10492 **7.3.2.16.27.1 Operation Event Sources**

10493 This field indicates where the event was triggered from.

10494 **Table 7-28. Operation Event Source Value**

Value	Source
0x00	Keypad
0x01	RF
0x02	Manual
0x03	RFID
0xFF	Indeterminate

10495 **7.3.2.16.27.2 Operation Event Codes**

<sup>114</sup> CCB 2430 PIN/RFID type is Octet String (octstr)

10496 The door lock optionally sends out notifications (if they are enabled) whenever there is a significant operational event  
 10497 on the lock. When combined with a source from the Event Source table above, the following operational event codes  
 10498 constitute an event on the door lock that can be both logged and sent to a bound device using the Operation Event  
 10499 Notification command.

10500 Not all operation event codes are applicable to each of the event source. Table 7-29 marks each event code with “A”  
 10501 if the event code is applicable to the event source.

10502 **Table 7-29. Operation Event Code Value**

Value	Operation Event Code	Keypad	RF	Manual	RFID
0x00	UnknownOrMfgSpecific	A	A	A	A
0x01	Lock	A	A	A	A
0x02	Unlock	A	A	A	A
0x03	LockFailureInvalidPINorID	A	A		A
0x04	LockFailureInvalidSchedule	A	A		A
0x05	UnlockFailureInvalidPINorID	A	A		A
0x06	UnlockFailureInvalidSchedule	A	A		A
0x07	OneTouchLock			A	
0x08	KeyLock			A	
0x09	KeyUnlock			A	
0x0A	AutoLock			A	
0x0B	ScheduleLock			A	
0x0C	ScheduleUnlock			A	
0x0D	Manual Lock (Key or Thumbturn)			A	
0x0E	Manual Unlock (Key or Thumbturn)			A	
0x0F	Non-Access User Operational Event	A			

10503 **7.3.2.16.27.3 User ID**

10504 The User ID who performed the event.

10505 **7.3.2.16.27.4 PIN**

10506 The PIN that is associated with the User ID who performed the event.

10507 **7.3.2.16.27.5 LocalTime**

10508 The LocalTime that indicates when the event is triggered. If time is not supported, the field SHALL be populated with  
 10509 default not used value 0xFFFFFFFF.

10510 **7.3.2.16.27.6 Data**

10511 The operation event notification command contains a variable string, which can be used to pass data associated with  
 10512 a particular event. Generally this field will be left empty. However, manufacturer can choose to use this field to  
 10513 store/display manufacturer-specific information.

10514 **7.3.2.16.27.7 Keypad Operation Event Notification**

10515 Keypad Operation Event Notification feature is enabled by setting the associated bitmasks in the [Keypad Operation  
 10516 Event Mask attribute].

10517 **Table 7-30. Keypad Operation Event Value**

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x00	0x00	BIT(0)	Unknown or manufacturer-specific keypad operation event
0x00	0x01	BIT(1)	Lock, source: keypad
0x00	0x02	BIT(2)	Unlock, source: keypad
0x00	0x03	BIT(3)	Lock, source: keypad, error: invalid PIN
0x00	0x04	BIT(4)	Lock, source: keypad, error: invalid schedule
0x00	0x05	BIT(5)	Unlock, source: keypad, error: invalid code
0x00	0x06	BIT(6)	Unlock, source: keypad, error: invalid schedule
0x00	0x0F	BIT(7)	Non-Access User operation event, source keypad.

10518 **7.3.2.16.27.8 RF Operation Event Notification**

10519 RF Operation Event Notification feature is enabled by setting the associated bitmasks in the [RF Operation Event  
 10520 Mask attribute].

10521 **Table 7-31. RF Operation Event Value**

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x01	0x00	BIT(0)	Unknown or manufacturer-specific RF operation event
0x01	0x01	BIT(1)	Lock, source: RF
0x01	0x02	BIT(2)	Unlock, source: RF
0x01	0x03	BIT(3)	Lock, source: RF, error: invalid code
0x01	0x04	BIT(4)	Lock, source: RF, error: invalid schedule
0x01	0x05	BIT(5)	Unlock, source: RF, error: invalid code

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x01	0x06	BIT(6)	Unlock, source: RF, error: invalid schedule

10522 **7.3.2.16.27.9 Manual Operation Event Notification**

10523 Manual Operation Event Notification feature is enabled by setting the associated bitmasks in the [Manual Operation  
10524 Event Mask attribute] attribute.

10525 **Table 7-32. Manual Operation Event Value**

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x02	0x00	BIT(0)	Unknown or manufacturer-specific manual operation event
0x02	0x01	BIT(1)	Thumbturn Lock
0x02	0x02	BIT(2)	Thumbturn Unlock
0x02	0x07	BIT(3)	One touch lock
0x02	0x08	BIT(4)	Key Lock
0x02	0x09	BIT(5)	Key Unlock
0x02	0x0A	BIT(6)	Auto lock
0x02	0x0B	BIT(7)	Schedule Lock
0x02	0x0C	BIT(8)	Schedule Unlock
0x02	0x0D	BIT(9)	Manual Lock (Key or Thumbturn)
0x02	0x0E	BIT(10)	Manual Unlock (Key or Thumbturn)

10526 **7.3.2.16.27.10 RFID Operation Event Notification**

10527 RFID Operation Event Notification feature is enabled by setting the associated bitmasks in the [RFID Operation Event  
10528 Mask attribute].

10529 **Table 7-33. RFID Operation Event Value**

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x03	0x00	BIT(0)	Unknown or manufacturer-specific keypad operation event
0x03	0x01	BIT(1)	Lock, source: RFID
0x03	0x02	BIT(2)	Unlock, source: RFID

Event Source	Operation Event Code	Attribute Bitmask	Event Description
0x03	0x03	BIT(3)	Lock, source: RFID, error: invalid RFID ID
0x03	0x04	BIT(4)	Lock, source: RFID, error: invalid schedule
0x03	0x05	BIT(5)	Unlock, source: RFID, error: invalid RFID ID
0x03	0x06	BIT(6)	Unlock, source: RFID, error: invalid schedule

10530 **7.3.2.16.28 Programming Event Notification Command**

10531 The door lock server sends out a programming event notification whenever a programming event takes place on the  
 10532 door lock.

10533 As with operational events, all programming events can be turned on and off by flipping bits in the associated event  
 10534 mask.

10535 The programming event notification command includes an optional string of data that can be used by the manufacturer  
 10536 to pass some manufacturer-specific information if that is required.

10537 **Figure 7-54. Format of the Programming Event Notification Command**

Octets	1	1	2	1	1	1	4	Variable/0
Data Type	uint8	uint8	uint16	octstr <sup>115</sup>	enum8	uint8	uint32	string
Field Name	Program Event Source	Program Event Code	User ID	PIN	User Type	User Status	LocalTime	Data

10538 **7.3.2.16.28.1 Operation Event Sources**

10539 This field indicates where the event was triggered from.

10540 **Table 7-34. Operation Event Source Value**

Value	Source
0x00	Keypad
0x01	RF
0x02	Reserved (Manual in Operation Event)
0x03	RFID
0xFF	Indeterminate

10541 **7.3.2.16.28.2 Programming Event Codes**

<sup>115</sup> CCB 2430 PIN/RFID type is Octet String (octstr)

10542 The door lock optionally sends out notifications (if they are enabled) whenever there is a significant programming  
10543 event on the lock. When combined with a source from the Event Source table above, the following programming event  
10544 codes constitute an event on the door lock that can be both logged and sent to a bound device using the Programming  
10545 Event Notification command.

10546 Not all event codes are applicable to each of the event source. Table 7-35 marks each event code with “A” if the event  
10547 code is applicable to the event source.

10548 **Table 7-35. Programming Event Codes**

Value	Programming Event Code	Keypad	RF	RFID
0x00	UnknownOrMfgSpecific	A	A	A
0x01	MasterCodeChanged	A		
0x02	PINCodeAdded	A	A	
0x03	PINCodeDeleted	A	A	
0x04	PINCodeChanged	A	A	
0x05	RFIDCodeAdded			A
0x06	RFIDCodeDeleted			A

10549 **7.3.2.16.28.3 User ID**

10550 The User ID who performed the event

10551 **7.3.2.16.28.4 PIN**

10552 The PIN that is associated with the User ID who performed the event

10553 **7.3.2.16.28.5 User Type**

10554 The User Type that is associated with the User ID who performed the event

10555 **7.3.2.16.28.6 User Status**

10556 The User Status that is associated with the User ID who performed the event

10557 **7.3.2.16.28.7 LocalTime**

10558 The LocalTime that indicates when the event is triggered. If time is not supported, the field SHALL be populated with  
10559 default not used value 0xFFFFFFFF.

10560 **7.3.2.16.28.8 Data**

10561 The programming event notification command contains a variable string, which can be used to pass data associated  
10562 with a particular event. Generally this field will be left empty. However, manufacturer can choose to use this field to  
10563 store/display manufacturer-specific information.

10564 **7.3.2.16.28.9 Keypad Programming Event Notification**

10565 Keypad Programming Event Notification feature is enabled by setting the associated bitmasks in the [Keypad  
10566 Programming Event Mask attribute].

10567

**Table 7-36. Keypad Programming Event Value**

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x00	0x00	BIT(0)	Unknown or manufacturer-specific keypad programming event
0x00	0x01	BIT(1)	Master code changed, source: keypad User ID: master user ID. PIN: default or master code if codes can be sent over the air per attribute. User type: default User Status: default
0x00	0x02	BIT(2)	PIN added, source: keypad User ID: user ID that was added. PIN: code that was added (if codes can be sent over the air per attribute.) User type: default or type added. User Status: default or status added.
0x00	0x03	BIT(3)	PIN deleted, source: keypad User ID: user ID that was deleted. PIN: code that was deleted (if codes can be sent over the air per attribute.) User type: default or type deleted. User Status: default or status deleted.
0x00	0x04	BIT(4)	PIN changed Source: keypad User ID: user ID that was changed PIN: code that was changed (if codes can be sent over the air per attribute.) User type: default or type changed. User Status: default or status changed.

10568 **7.3.2.16.28.10 RF Programming Event Notification**

10569 RF Programming Event Notification feature is enabled by setting the associated bitmasks in the [RF Programming  
 10570 Event Mask attribute].

10571

**Table 7-37. RF Programming Event Value**

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x01	0x00	BIT(0)	Unknown or manufacturer-specific RF programming event.
0x01	0x02	BIT(2)	PIN added, source RF Same as keypad source above

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x01	0x03	BIT(3)	PIN deleted, source RF Same as keypad source above.
0x01	0x04	BIT(4)	PIN changed Source RF Same as keypad source above
0x01	0x05	BIT(5)	RFID code added, Source RF
0x01	0x06	BIT(6)	RFID code deleted, Source RF

10572 **7.3.2.16.28.11 RFID Programming Event Notification**

10573 RFID Programming Event Notification feature is enabled by setting the associated bitmasks in the [RFID  
10574 Programming Event Mask attribute].

10575 **Table 7-38. RFID Programming Event Value**

Event Source	Program Event Code	Attribute Bitmask	Event Description
0x03	0x00	BIT(0)	Unknown or manufacturer-specific keypad programming event
0x03	0x05	BIT(5)	ID Added, Source: RFID User ID: user ID that was added. ID: ID that was added (if codes can be sent over the air per attribute.) User Type: default or type added. User Status: default or status added.
0x03	0x06	BIT(6)	ID Deleted, Source: RFID User ID: user ID that was deleted. ID: ID that was deleted (if codes can be sent over the air per attribute.) User Type: default or type deleted. User Status: default or status deleted.

10576 **7.3.2.17 Scene Table Extension**

10577 If the Scene server cluster is implemented, the following extension field is added to the Scene table:

10578 • **LockState**

10579 When the *LockState* attribute is part of a Scene table, the attribute is treated as a writable command; that is,  
10580 setting the *LockState* to lock will command the lock to lock, and setting the *LockState* to unlock will command  
10581 the lock to unlock. Setting the *LockState* attribute to “not fully locked” is not supported.

10582 The Transition Time field in the Scene table will be treated as a delay before setting the *LockState* attribute; that  
10583 is, it is possible to activate a scene with the lock actuation some seconds later.



10584 Locks that do not have an actuation mechanism SHOULD not support the Scene table extension.

### 10585 7.3.3 Client

10586 The client supports no cluster specific attributes. The client receives the cluster-specific commands generated by the  
 10587 server, as shown in Table 7-27. The client generates the cluster-specific commands that will be received by the server,  
 10588 as shown in Table 7-23.

## 10589 7.4 Window Covering

### 10590 7.4.1 Overview

10591 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 10592 etc.

10593 The window covering cluster provides an interface for controlling and adjusting automatic window coverings such as  
 10594 drapery motors, automatic shades, and blinds.

#### 10595 7.4.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCB 1994 1995 1996 1997 2086 2094 2095 2096 2097
2	CCB 2328

#### 10596 7.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	WNCV	Type 1 (client to server)

#### 10597 7.4.1.3 Cluster Identifiers

Identifier	Name
0x0102	Window Covering

### 10598 7.4.2 Server

#### 10599 7.4.2.1 Attributes

10600 For convenience, the attributes defined in this cluster are arranged into sets of related attributes; each set can contain  
 10601 up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the attribute  
 10602 set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets are listed  
 10603 in Table 7-39.

10604

**Table 7-39. Window Covering Attribute Set**

Attribute Set Identifier	Description
0x00	Window Covering Information
0x01	Window Covering Settings

10605 **7.4.2.1.1 Window Covering Information Attribute Set**

10606 The Window Covering Information attribute set contains the attributes summarized in Table 7-40.

10607

**Table 7-40. Window Covering Information Attribute Set**

Id	Name	Unit	Type	Range	Acc	Default	M/O
0x0000	<i>WindowCoveringType</i>		enum8	0x00 – 0x09	R	0x00	M
0x0001	<i>PhysicalClosedLimit – Lift</i>	cm	uint16	0x0000 – 0xffff	R	0x0000	O
0x0002	<i>PhysicalClosedLimit – Tilt</i>	0.1°	uint16	0x0000 – 0xffff	R	0x0000	O
0x0003	<i>CurrentPosition – Lift</i>	cm	uint16	0x0000 – 0xffff	R	0x0000	O
0x0004	<i>Current Position – Tilt</i>	0.1°	uint16	0x0000 – 0xffff	R	0x0000	O
0x0005	<i>Number of Actuations – Lift</i>		uint16	0x0000 – 0xffff	R	0x0000	O
0x0006	<i>Number of Actuations – Tilt</i>		uint16	0x0000 – 0xffff	R	0x0000	O
0x0007	<i>Config/Status</i>		map8	0xxx xxxx	R	0000 0011	M
0x0008	<i>Current Position Lift Percentage</i>		uint8	0-0x64	RSP	0x00	M*
0x0009	<i>Current Position Tilt Percentage</i>		uint8	0-0x64	RSP	0x00	M*

10608

10609 **\*Note:** "M\*" designates that the related attributes are required to be mandatory only if Closed Loop  
 10610 control is enabled and Lift/Tilt actions are correspondingly supported, i.e. the  
 10611 *CurrentPositionLiftPercentage* attribute SHALL be mandatory only if Closed Loop control and Lift actions  
 10612 are supported; and the *CurrentPositionTiltPercentage* attribute SHALL be mandatory only if Closed Loop  
 10613 control and Tilt actions are supported.

10614 **7.4.2.1.2 WindowCoveringType Attribute**

10615 The *WindowCoveringType* attribute identifies the type of window covering being controlled by this endpoint and  
 10616 SHALL be set to one of the non-reserved values in Table 7-41.

10617

**Table 7-41. Window Covering Type**

Value	Window Covering Type	Supported Actions
0x00	Rollershade	Lift
0x01	Rollershade - 2 Motor	Lift
0x02	Rollershade – Exterior	Lift
0x03	Rollershade - Exterior - 2 Motor	Lift
0x04	Drapery	Lift
0x05	Awning	Lift
0x06	Shutter	Tilt
0x07	Tilt Blind - Tilt Only	Tilt
0x08	Tilt Blind - Lift and Tilt	Lift, Tilt
0x09	Projector Screen	Lift

10618 **7.4.2.1.2.1 PhysicalClosedLimit - Lift Attribute**

10619 The *PhysicalClosedLimitLift* attribute identifies the maximum possible encoder position possible (in centimeters) to  
 10620 position the height of the window covering – this is ignored if the device is running in Open Loop Control.

10621 **7.4.2.1.2.2 PhysicalClosedLimit - Tilt Attribute**

10622 The *PhysicalClosedLimitTilt* attribute identifies the maximum possible encoder position possible (tenth of a degrees)  
 10623 to position the angle of the window covering – this is ignored if the device is running in Open Loop Control.

10624 **7.4.2.1.2.3 CurrentPosition - Lift Attribute**

10625 The *CurrentPositionLift* attribute identifies the actual position (in centimeters) of the window covering from the top  
 10626 of the shade if Closed Loop Control is enabled. This attribute is ignored if the device is running in Open Loop Control.

10627 **7.4.2.1.2.4 Current Position - Tilt Attribute**

10628 The *CurrentPositionTilt* attribute identifies the actual tilt position (in tenth of an degree) of the window covering from  
 10629 Open if Closed Loop Control is enabled. This attribute is ignored if the device is running in Open Loop Control.

10630 **7.4.2.1.2.5 Number of Actuations - Lift Attribute**

10631 The *NumberOfActuationsLift* attribute identifies the total number of lift actuations applied to the Window Covering  
 10632 since the device was installed.

10633 **7.4.2.1.2.6 Number of Actuations - Tilt Attribute**

10634 The *NumberOfActuationsTilt* attribute identifies the total number of tilt actuations applied to the Window Covering  
 10635 since the device was installed.

10636 **7.4.2.1.2.7 Config/Status Attribute**

10637 The *ConfigStatus* attribute makes configuration and status information available. To change settings, devices SHALL  
 10638 write to the Mode attribute of the Window Covering Settings Attribute Set. The behavior causing the setting or clearing  
 10639 of each bit is vendor specific. See Table 7-42 for details on each bit.

10640

**Table 7-42. Bit Meanings for the Config/Status Attribute**

Bit	Meaning	Description
bit0	0 = Not Operational 1 = Operational	Operational: This status bit defines if the Window Covering is operational.
bit1	0 = Not Online 1 = Online	Online: This status bit defines if the Window Covering is enabled for transmitting over the network.
bit2	0 = Commands are normal 1 = Open/Up Commands have been reversed	Reversal – Lift commands: This status bit identifies if the direction of rotation for the Window Covering has been reversed in order for Open/Up commands to match the physical installation condition.
bit3	0 = Lift control is Open Loop 1 = Lift control is Closed Loop	Control – Lift: This status bit identifies if the window covering supports Open Loop or Closed Loop Lift Control
bit4	0 = Tilt control is Open Loop 1 = Tilt control is Closed Loop	Control – Tilt: This status bit identifies if the window covering supports Open Loop or Closed Loop Tilt Control
bit5	0 = Timer Controlled 1 = Encoder Controlled This bit is Ignored if running Lift in Open Loop Control.	Encoder – Lift: This status bit identifies if a Closed Loop Controlled Window Covering is employing an encoder for positioning the height of the window covering.
bit6	0 = Timer Controlled 1 = Encoder Controlled This bit is Ignored if running Tilt in Open Loop Control.	Encoder – Tilt: This status bit identifies if a Closed Loop Controlled Window Covering is employing an encoder for tilting the window covering.

#### 10641 7.4.2.1.2.8 Current Position Lift Percentage Attribute

10642 The *CurrentPositionLiftPercentage* attribute identifies the actual position as a percentage between the  
 10643 *InstalledOpenLimitLift* attribute and the *InstalledClosedLimitLift* attribute of the window covering from the up/open  
 10644 position if Closed Loop Control is enabled. If the device is running in Open Loop Control or the device only supports  
 10645 Tilt actions, this attribute is not required as an attribute but has a special interpretation when received as part of a scene  
 10646 command (see “Scene Table Extensions” below).

#### 10647 7.4.2.1.2.9 Current Position Tilt Percentage Attribute

10648 The *CurrentPositionTiltPercentage* attribute identifies the actual position as a percentage between the  
 10649 *InstalledOpenLimitTilt* attribute and the *InstalledClosedLimitTilt* attribute of the window covering from the up/open  
 10650 position if Closed Loop Control is enabled. If the device is running in Open Loop Control or the device only support  
 10651 Lift actions, this attribute is not required as an attribute but has a special interpretation when received as part of a scene  
 10652 command (see “Scene Table Extensions” below).

10653 **7.4.2.1.3 Window Covering Settings Attribute Set**

10654 The *WindowCoveringSettings* attribute set contains the attributes summarized in Table 7-43.

10655 **Table 7-43. Window Covering Settings Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Unit</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Default</b>	<b>M/O</b>
0x0000	<i>InstalledOpenLimit – Lift</i>	<i>cm</i>	uint16	0x0000 – 0xffff	R	0x0000	M*
0x0001	<i>InstalledClosedLimit – Lift</i>	<i>cm</i>	uint16	0x0000 – 0xffff	R	0xffff	M*
0x0002	<i>InstalledOpenLimit – Tilt</i>	<i>0.1°</i>	uint16	0x0000 – 0xffff	R	0x0000	M*
0x0003	<i>InstalledClosedLimit – Tilt</i>	<i>0.1°</i>	uint16	0x0000 – 0xffff	R	0xffff	M*
0x0004	<i>Velocity – Lift</i>	<i>cm/sec</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0005	<i>Acceleration Time – Lift</i>	<i>0.1sec</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0006	<i>Deceleration Time – Lift</i>	<i>0.1sec</i>	uint16	0x0000 – 0xffff	RW	0x0000	O
0x0007	<i>Mode</i>		map8		RW	0000 <sup>116</sup> 0100	M
0x0008	<i>Intermediate Setpoints – Lift</i>		octstr	-	RW	“1,0x0000”	O
0x0009	<i>Intermediate Setpoints – Tilt</i>		octstr	-	RW	“1,0x0000”	O

10656

10657 **\*Note:** “M\*” designates that the related attributes are required to be mandatory only if Closed Loop  
 10658 control is enabled and Lift/Tilt actions are correspondingly supported, i.e. the *InstalledOpenLimitLift* and  
 10659 *InstalledClosedLimitLift* attributes SHALL be mandatory only if Closed Loop control and Lift actions are  
 10660 supported; and the *InstalledOpenLimitTilt* and *InstalledClosedLimitTilt* attributes SHALL be mandatory  
 10661 only if Closed Loop control and Tilt actions are supported.

10662 **7.4.2.1.3.1 InstalledOpenLimit – Lift**

10663 The *InstalledOpenLimitLift* attribute identifies the Open Limit for Lifting the Window Covering whether position (in  
 10664 centimeters) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only  
 10665 supports Tilt actions.

10666 **7.4.2.1.3.2 InstalledClosedLimit – Lift**

10667 The *InstalledClosedLimitLift* attribute identifies the Closed Limit for Lifting the Window Covering whether position  
 10668 (in centimeters) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only  
 10669 supports Tilt actions.

10670 **7.4.2.1.3.3 InstalledOpenLimit – Tilt**

<sup>116</sup> CCB 2328 no range for map and fix default

10671 The *InstalledOpenLimitTilt* attribute identifies the Open Limit for Tilting the Window Covering whether position (in  
10672 tenth of a degree) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or only  
10673 supports Lift actions.

#### 10674 **7.4.2.1.3.4 InstalledClosedLimit – Tilt**

10675 The *InstalledClosedLimitTilt* attribute identifies the Closed Limit for Tilting the Window Covering whether position  
10676 (in tenth of a degree) is encoded or timed. This attribute is ignored if the device is running in Open Loop Control or  
10677 only supports Lift actions.

#### 10678 **7.4.2.1.3.5 Velocity – Lift**

10679 The *VelocityLift* attribute identifies the velocity (in centimeters per second) associated with Lifting the Window  
10680 Covering.

#### 10681 **7.4.2.1.3.6 Acceleration Time – Lift**

10682 The *AccelerationTimeLift* attribute identifies any ramp up times to reaching the velocity setting (in tenth of a second)  
10683 for positioning the Window Covering.

#### 10684 **7.4.2.1.3.7 Deceleration Time – Lift**

10685 The *DecelerationTimeLift* attribute identifies any ramp down times associated with stopping the positioning (in tenth  
10686 of a second) of the Window Covering.

#### 10687 **7.4.2.1.3.8 Mode**

10688 The *Mode* attribute allows configuration of the Window Covering, such as: reversing the motor direction, placing the  
10689 Window Covering into calibration mode, placing the motor into maintenance mode, disabling the network, and  
10690 disabling status LEDs. See Table 7-44 for details.

10691 **Table 7-44. Bit Meanings for the Mode Attribute**

Bit	Meaning	Description
bit0	0 = motor direction is normal 1 = motor direction is reversed	Disables (0) or Enables (1) the reversal of the motor rotating direction associated with an UP/OPEN command. Should be set so that an UP/OPEN command matches moving the Window Covering physically in that direction.
bit1	0 = run in normal mode 1 = run in calibration mode	Disables (0) or Enables (1) placing the Window Covering into Calibration Mode where limits are either setup using physical tools or limits are learned by the controller based on physical setup of the Window Covering by an installer.
bit2	0 = motor is running normally 1 = motor is running in maintenance mode	Disables (0) or Enables (1) placing the motor into Maintenance Mode where the motor cannot be moved over the network or by a switch connected to a Local Switch Input.
bit3	0 = LEDs are off 1 = LEDs will display feedback	Disables (0) or Enables (1) the display of any feedback LEDs resident especially on the packaging of an endpoint where they may cause distraction to the occupant.

#### 10692 **7.4.2.1.3.9 Intermediate Setpoints – Lift**

10693 Identifies the number of Intermediate Setpoints supported by the Window Covering for Lift and then identifies the  
 10694 position settings for those Intermediate Setpoints if Closed Loop Control is supported. This is a comma delimited  
 10695 ASCII character string. For example: “2,0x0013, 0x0030”

10696 **7.4.2.1.3.10 Intermediate Setpoints – Tilt**

10697 Identifies the number of Intermediate Setpoints supported by the Window Covering for Tilt and then identifies the  
 10698 position settings for those Intermediate Setpoints if Closed Loop Control is supported. This is a comma delimited  
 10699 ASCII character string. For example: “2,0x0013, 0x0030”

10700

10701 **7.4.2.2 Commands Received**

10702 **Table 7-45. Commands Received by the Window Covering Server Cluster**

Command ID	Description	M/O
0x00	Up / Open	M
0x01	Down / Close	M
0x02	Stop	M
0x04	Go To Lift Value	O
0x05	Go to Lift Percentage	O
0x07	Go to Tilt Value	O
0x08	Go to Tilt Percentage	O

10703 **7.4.2.2.1 Up / Open Command**

10704 **7.4.2.2.1.1 Payload Format**

10705 This command has no payload.

10706 **7.4.2.2.1.2 Effect on Receipt**

10707 Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the  
 10708 *InstalledOpenLimit – Lift* and the tilt is at the *InstalledOpenLimit – Tilt*. This will happen as fast as possible.

10709 **7.4.2.2.2 Down / Close Command**

10710 **7.4.2.2.2.1 Payload Format**

10711 This command has no payload.

10712 **7.4.2.2.2.2 Effect on Receipt**

10713 Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the  
 10714 *InstalledClosedLimit – Lift* and the tilt is at the *InstalledClosedLimit – Tilt*. This will happen as fast as possible.

**10715 7.4.2.2.3 Stop Command****10716 7.4.2.2.3.1 Payload Format**

10717 This command has no payload.

**10718 7.4.2.2.3.2 Effect on Receipt**

10719 Upon receipt of this command, the Window Covering will stop any adjusting to the physical tilt and lift that is currently  
10720 occurring.

**10721 7.4.2.2.4 Go To Lift Value****10722 7.4.2.2.4.1 Payload Format**

10723 The Go To Lift Value command payload SHALL be formatted as illustrated in Figure 7-55.

10724 **Figure 7-55. Format of the Go To Lift Value Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Lift Value

**10725 7.4.2.2.4.2 Effect on Receipt**

10726 Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the lift value  
10727 specified in the payload of this command as long as that value is not larger than *InstalledOpenLimit – Lift* and not  
10728 smaller than *InstalledClosedLimit – Lift*. If the lift value is out of bounds a default response containing the status of  
10729 INVALID\_VALUE will be returned.

**10730 7.4.2.2.5 Go to Lift Percentage****10731 7.4.2.2.5.1 Payload Format**

10732 The Go To Lift Percentage command payload SHALL be formatted as illustrated in Figure 7-56.

10733 **Figure 7-56. Format of the Go To Lift Percentage Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Percentage Lift Value

**10734 7.4.2.2.5.2 Effect on Receipt**

10735 Upon receipt of this command, the Window Covering will adjust the window so the physical lift is at the lift percentage  
10736 specified in the payload of this command. The percentage value will be mapped to a 8-bit unsigned integer value  
10737 between *InstalledOpenLimit* and *InstalledClosedLimit*. If the percentage lift value is larger than 100, no physical  
10738 action will be taken and a default response containing the status of INVALID\_VALUE will be returned. If the device  
10739 only supports open loop lift action then a zero percentage SHOULD be treated as a down/close command and a non-  
10740 zero percentage SHOULD be treated as an up/open command. If the device is only a tilt control device, then the  
10741 command SHOULD be ignored and a UNSUPPORTED\_COMMAND status SHOULD be returned. The device must  
10742 support either the Go To Lift Percentage or the Go To Tilt Percentage command.



10743 **7.4.2.2.6 Go to Tilt Value**

10744 **7.4.2.2.6.1 Payload Format**

10745 The Go To Tilt Value command payload SHALL be formatted as illustrated in Figure 7-57.

10746 **Figure 7-57. Format of the Go To Tilt Value Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Tilt Value

10747 **7.4.2.2.6.2 Effect on Receipt**

10748 Upon receipt of this command, the Window Covering will adjust the window so the physical tilt is at the tilt value  
 10749 specified in the payload of this command as long as that value is not larger than *InstalledOpenLimit – Tilt* and not  
 10750 smaller than *InstalledClosedLimit – Tilt*. If the tilt value is out of bounds a default response containing the status of  
 10751 INVALID\_VALUE will be returned.

10752 **7.4.2.2.7 Go to Tilt Percentage**

10753 **7.4.2.2.7.1 Payload Format**

10754 The Go To Tilt Percentage command payload SHALL be formatted as illustrated below.

10755 **Figure 7-58. Format of the Go To Lift Percentage Command**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Percentage Tilt Value

10756 **7.4.2.2.7.2 Effect on Receipt**

10757 Upon receipt of this command, the Window Covering will adjust the window so the physical tilt is at the tilt percentage  
 10758 specified in the payload of this command. The percentage value will be mapped to a 8-bit unsigned integer value  
 10759 between *InstalledOpenLimit-Tilt* and *InstalledClosedLimit-Tilt*. If the percentage tilt value is larger than 100, no  
 10760 physical action will be taken and a default response containing the status of INVALID\_VALUE will be returned. If  
 10761 the device only supports open loop tilt action then a zero percentage SHOULD be treated as a down/close command  
 10762 and a non-zero percentage SHOULD be treated as an up/open command. If the device is only a lift control device,  
 10763 then the command SHOULD be ignored and a UNSUPPORTED\_COMMAND status SHOULD be returned. The  
 10764 device must support either the Go To Lift Percentage or the Go To Tilt Percentage command.

10765 **7.4.2.3 Commands Generated**

10766 This cluster uses the standard Default Response command defined in the ZCL specification for responding to received  
 10767 commands. Possible status values that can be returned are: SUCCESS, NOT\_FOUND, NOT\_AUTHORIZED,  
 10768 INSUFFICIENT\_SPACE, UNSUP\_CLUSTER\_COMMAND, INVALID\_FIELD, INVALID\_VALUE,  
 10769 HARDWARE\_FAILURE, FAILURE.

10770 **7.4.2.4 Scene Table Extensions**

10771 If the Window Covering server cluster is implemented, the following extension field is added to the Scene table:

10772 • **CurrentPositionLiftPercentage**  
10773 When the *CurrentPositionLiftPercentage* attribute is part of a Scene table, the attribute is treated as a writeable  
10774 command, that is, setting the lift percentage of the covering device to the value specified in the scene table  
10775 extension over the specified transition time. The device MAY treat the command as a linear transition if  
10776 appropriate or MAY accelerate and decelerate as it deems necessary. If the device is only a tilt controlling  
10777 device this scene table extension is ignored. If the device is an open loop controlled lift device, then a  
10778 percentage of 0 is treated as a close command and a non zero percentage is treated as an open command and the  
10779 device will ignore the transition time and transition as fast as appropriate for that device.

10780 • **CurrentPositionTiltPercentage**  
10781 When the *CurrentPositionTiltPercentage* attribute is part of a Scene table, the attribute is treated as a writeable  
10782 command, that is, setting the tilt percentage of the covering device to the value specified in the scene table  
10783 extension over the specified transition time. The device MAY treat the command as a linear transition if  
10784 appropriate or MAY accelerate and decelerate as it deems necessary. If the device is only a lift controlling  
10785 device this scene table extension is ignored. If the device is an open loop controlled tilt device, then a  
10786 percentage of 0 is treated as a close command and a non zero percentage is treated as an open command and the  
10787 device will ignore the transition time and transition as fast as appropriate for that device.

#### 10788 **7.4.2.5 Attribute Reporting**

10789 This cluster SHALL support attribute reporting using the Report Attributes command and according to the minimum  
10790 and maximum reporting interval settings described in the ZCL. The following attributes SHALL be reported:

10791 *Current Position - Lift Percentage*

10792 *Current Position - Tilt Percentage*

### 10793 **7.4.3 Client**

10794 The client has no cluster specific attributes. No cluster specific commands are received by the client. The client  
10795 generates the cluster specific commands detailed in sub-clause 7.4.2.2.

## CHAPTER 8 SECURITY AND SAFETY

10796

10797 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
10798 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
10799 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
10800 using [*Rn*] notation.

### 8.1 General Description

#### 8.1.1 Introduction

10802 The clusters specified in this document are for use in security and safety related applications.

10804 The clusters currently defined are those that are used by wireless Intruder Alarm Systems (IAS). Intruder Alarm  
10805 systems include functions for the detection of intruders and/or triggering, processing of information, notification of  
10806 alarms and the means to operate the IAS.

10807 Functions additional to those MAY be included in IAS providing they do not influence the correct operation of the  
10808 mandatory functions. Components of other applications MAY be combined or integrated with a IAS, providing the  
10809 performance of the IAS components is not adversely influenced.

#### 8.1.2 Cluster List

10810 This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of  
10811 clarification.  
10812

10813 The clusters defined in this document are listed in Table 8-1.

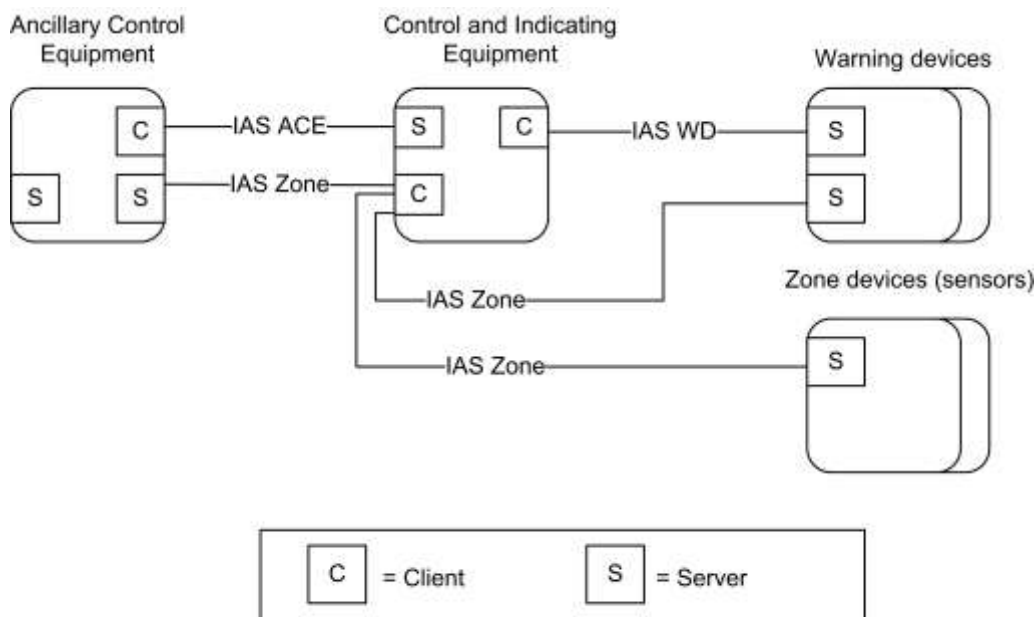
10814 **Table 8-1. Clusters of the Security and Safety Functional Domain**

Cluster ID	Cluster Name	Description
0x500	IAS Zone	Attributes and commands for IAS security zone devices.
0x501	IAS ACE	Attributes and commands for IAS Ancillary Control Equipment.
0x502	IAS WD	Attributes and commands for IAS Warning Devices

10815

10816

Figure 8-1. Typical Usage of the IAS Clusters



Note: Device names are examples for illustration purposes only

10817

## 10818 8.2 IAS Zone

### 10819 8.2.1 Overview

10820 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

10822 The IAS Zone cluster defines an interface to the functionality of an IAS security zone device. IAS Zone supports up to two alarm types per zone, low battery reports and supervision of the IAS network.

#### 10824 8.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; ZHA 1.2 and 1.2.1 features; CCB 2044 2045
2	CCB 2352

#### 10825 8.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASZ	Type 2 (server to client)

10826 **8.2.1.3 Cluster Identifiers**

Identifier	Name
0x0500	IAS Zone

10827 **8.2.2 Server**

10828 **8.2.2.1 Attributes**

10829 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 10830 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 10831 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 10832 are listed in Table 8-2.

10833 **Table 8-2. Attribute Sets for the IAS Zone Cluster**

Attribute Set Identifier	Description
0x000	Zone information
0x001	Zone settings

10834 **8.2.2.1.1 Zone Information Attribute Set**

10835 The Zone Information attribute set contains the attributes summarized in Table 8-3.

10836 **Table 8-3. Attributes of the Zone Information Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>ZoneState</i>	enum8	<i>All</i>	R	0x00	M
0x0001	<i>ZoneType</i>	enum16	<i>All</i>	R	-	M
0x0002	<i>ZoneStatus</i>	map16	<i>All</i>	R	0x00	M

10837 **8.2.2.1.1.1 ZoneState Attribute**

10838 The *ZoneState* attribute contains the values summarized in Table 8-4.

10839 **Table 8-4. Values of the ZoneState Attribute**

Attribute Value	Meaning
0x00	Not enrolled
0x01	Enrolled (the client will react to Zone State Change Notification commands from the server)

10840 **8.2.2.1.1.2 ZoneType Attribute**

10841 The *ZoneType* attribute values are summarized in Table 8-5. The Zone Type dictates the meaning of Alarm1 and  
10842 Alarm2 bits of the *ZoneStatus* attribute, as also indicated in this table.

10843 **Table 8-5. Values of the *ZoneType* Attribute**

Value	Zone Type	Alarm1	Alarm2
0x0000	Standard CIE	System Alarm	-
0x000d	Motion sensor	Intrusion indication	Presence indication
0x0015	Contact switch	1 <sup>st</sup> portal Open-Close	2 <sup>nd</sup> portal Open-Close
0x0016	Door/Window handle <sup>117</sup>	See Table 8-7	See Table 8-7
0x0028	Fire sensor	Fire indication	-
0x002a	Water sensor	Water overflow indication	-
0x002b	Carbon Monoxide (CO) sensor	CO indication	Cooking indication
0x002c	Personal emergency device	Fall/Concussion	Emergency button
0x002d	Vibration/Movement sensor	Movement indication	Vibration
0x010f	Remote Control	Panic	Emergency
0x0115	Key fob	Panic	Emergency
0x021d	Keypad	Panic	Emergency
0x0225	Standard Warning Device (see [N1] part 4)	-	-
0x0226	Glass break sensor	Glass breakage detected	-
0x0229	Security repeater*	-	-
0x8000-0xffff	manufacturer specific types	-	-
0xffff	Invalid Zone Type	-	-

10844 \* For example: a repeater for security devices that needs to be supervised by the alarm panel/IAS  
10845 CIE to ensure a reliable security sensor network

10846 **8.2.2.1.1.3 *ZoneStatus* Attribute**

10847 The *ZoneStatus* attribute is a bit map. The meaning of each bit is summarized in Table 8-6.

10848 **Table 8-6. Values of the *ZoneStatus* Attribute**

Bit	Meaning	Values
0	Alarm1	1 – opened or alarmed

<sup>117</sup> NFR Door/Window Position

Bit	Meaning	Values
		0 – closed or not alarmed
1	Alarm2	1 – opened or alarmed 0 – closed or not alarmed
2	Tamper	1 – Tampered 0 – Not tampered
3	Battery	1 – Low battery 0 – Battery OK
4	Supervision Notify	1 – Notify 0 – Does not notify
5	Restore Notify	1 – Notify restore 0 – Does not notify of restore
6	Trouble	1 – Trouble/Failure 0 – OK
7	AC (mains)	1 – AC/Mains fault 0 – AC/Mains OK
8	Test	1 – Sensor is in test mode 0 – Sensor is in operation mode
9	Battery Defect	1 – Sensor detects a defective battery 0 – Sensor battery is functioning normally

10849

10850 For the door/window handle zone type (0x0016), the Alarm1 and Alarm2 bits of the of the *ZoneStatus*  
 10851 attribute are to be interpreted as indicated below.<sup>118</sup>

10852

10853

**Table 8-7. Usage of alarm bits of *ZoneStatus* Attribute for *door/window handle zone type (0x0016)***

Alarm1	Alarm2	Description
0b0	0b0	(Door/window) closed
0b1	0b0	(Door/window) tilted (partly open)
0b1	0b1	(Door/window) open

10854 **8.2.2.1.1.3.1 Supervision Notify<sup>119</sup>**

10855 This bit indicates whether the Zone issues periodic Zone Status Change Notification commands. The CIE device MAY  
 10856 use these periodic notifications as an indication that a zone is operational. Zones that do not implement periodic  
 10857 notifications are required to set this bit to zero (the CIE will know not to interpret the lack of reports as a problem).

<sup>118</sup> NFR Door/Window Position

<sup>119</sup> CCB 2352 explain interval is not defined and clarify that notifications are not general reporting commands

10858 The notification interval is not configurable. It is manufacturer specific and is typically determined by local regulations  
10859 (e.g. UL requires a minimum of 28 mins). The Poll Control cluster SHOULD be used for sleeping end devices.

#### 10860 **8.2.2.1.1.3.2 Restore Notify<sup>120</sup>**

10861 This bit indicates whether or not a Zone Status Change Notification command will be sent to indicate that an alarm is  
10862 no longer present. Some Zones do not have the ability to detect that alarm condition is no longer present, they only  
10863 can tell that an alarm has occurred. These Zones must set the “Restore” bit to zero, indicating to the CIE not to look  
10864 for alarm-restore notifications.

### 10865 **8.2.2.1.2 Zone Settings Attribute Set**

10866 The Zone Settings attribute set contains the attributes summarized in Table 8-8.

10867 **Table 8-8. Attributes of the Zone Settings Attribute Set**

Id	Name	Type	Range	Access	Def	M/O
0x0010	<i>IAS_CIE_Address</i>	EUI64	-	RW	-	M
0x0011	<i>ZoneID</i>	uint8	0x00 – 0xFF	R	0xFF	M
0x0012	<i>NumberOfZoneSensitivityLevelsSupported</i>	uint8	0x02 – 0xff	R	0x02	O*
0x0013	<i>CurrentZoneSensitivityLevel</i>	uint8	0x00 – 0xff	RW	0x00	O*

10868 \* *These attributes depend on each other and if one is supported than both SHALL be supported.*

#### 10869 **8.2.2.1.2.1 IAS\_CIE\_Address Attribute**

10870 The *IAS\_CIE\_Address* attribute specifies the address that commands generated by the server SHALL be sent to. All  
10871 commands received by the server must also come from this address.

10872 It is up to the zone's specific implementation to permit or deny change (write) of this attribute at specific times.

10873 See section 8.2.2.1.3 for more information on setting this attribute.

#### 10874 **8.2.2.1.2.2 ZoneID Attribute**

10875 A unique reference number allocated by the CIE at zone enrollment time.

10876 Used by IAS devices to reference specific zones when communicating with the CIE. The *ZoneID* of each zone stays  
10877 fixed until that zone is unenrolled.

#### 10878 **8.2.2.1.2.3 NumberOfZoneSensitivityLevelsSupported Attribute**

10879 Provides the total number of sensitivity levels supported by the IAS Zone server. The purpose of this attribute is to  
10880 support devices that can be configured to be more or less sensitive (e.g., motion sensor). It provides IAS Zone clients  
10881 with the range of sensitivity levels that are supported so they MAY be presented to the user for configuration.

10882 The values 0x00 and 0x01 are reserved because a device that has zero or one sensitivity level SHOULD NOT support  
10883 this attribute because no configuration of the IAS Zone server's sensitivity level is possible.

10884 The meaning of each sensitivity level is manufacturer-specific. However, the sensitivity level of the IAS Zone server  
10885 SHALL become more sensitive as they ascend. For example, if the server supports three sensitivity levels, then the  
10886 value of this attribute would be 0x03 where 0x03 is more sensitive than 0x02, which is more sensitive than 0x01.

#### 10887 **8.2.2.1.2.4 CurrentZoneSensitivityLevel Attribute**

<sup>120</sup> CCB 2352 change from notes to body text



10888 Allows an IAS Zone client to query and configure the IAS Zone server’s sensitivity level. Please see Section  
10889 *NumberOfZoneSensitivityLevelsSupported* Attribute for more detail on how to interpret this attribute.

10890 The default value 0x00 is the device’s default sensitivity level as configured by the manufacturer. It MAY correspond  
10891 to the same sensitivity as another value in the *NumberOfZoneSensitivityLevelsSupported*, but this is the default  
10892 sensitivity to be used if the *CurrentZoneSensitivityLevel* attribute is not otherwise configured by an IAS Zone client.

### 10893 **8.2.2.1.3 Implementation Guidelines**

10894 Use of the *IAS\_CIE\_Address* and *ZoneID* attributes functions as an additional enrollment step that is not employed by  
10895 other devices. The reason for this is to provide an extra layer of security due to the nature of these devices in protecting  
10896 premises from physical intrusion and attack.

10897 There are three methods for enrolling IAS Zone server to an IAS CIE (i.e., IAS Zone client):

- 10898 • Trip-to-pair
- 10899 • Auto-Enroll-Response
- 10900 • Auto-Enroll-Request

10901 IAS Zone servers SHALL support either:

- 10902 • Trip-to-pair AND Auto-Enroll-Response, OR
- 10903 • Auto-Enroll-Request

10904 An IAS Zone client SHALL support either:

- 10905 • Trip-to-pair AND Auto-Enroll-Response, OR
- 10906 • Auto-Enroll-Request

10907 An IAS Zone client MAY support all enrollment methods. The Trip-to-Pair enrollment method is primarily intended  
10908 to be used when there is a desire for an explicit enrollment method (e.g., when a GUI wizard or other commissioning  
10909 tool is used by a user or installer to add multiple IAS Zone servers in an orderly fashion, assign names to them,  
10910 configure them in the system).

10911 A commissioning tool MAY act as an agent, on behalf of an IAS CIE device for either commissioning method. A  
10912 commissioning tool MAY perform any of the actions that are defined below for the IAS CIE.

10913 The following requirements are intended to ensure a timely and interoperable commissioning process:

- 10914 • After joining a network, an IAS Zone server implemented as a Sleepy End Device SHALL data poll at least  
10915 once every seven seconds until its *ZoneState* attribute has been updated to “enrolled” (i.e., until it receives a  
10916 Zone Enroll Response command from an IAS Zone client).
- 10917 • After joining a network, an IAS Zone server SHOULD data poll at least once every two seconds until its  
10918 *ZoneState* attribute has been updated to “enrolled” (i.e., until it receives a Zone Enroll Response command  
10919 from an IAS Zone client).
- 10920 • If the IAS Zone server supports Poll Control cluster, it SHOULD continue data polling at this rate until its  
10921 Poll Control cluster parameters are configured otherwise.
- 10922 • The *IAS\_CIE\_Address* attribute of the IAS Zone server to be enrolled SHALL be configured only by the  
10923 IAS CIE (or an agent of an IAS CIE). A self-configuration based on any kind of auto-detect approach  
10924 triggered by the IAS Zone server itself SHALL be prohibited.

10925 The detailed requirements for each commissioning method follow:

#### 10926 **Trip-to-Pair**

- 10927 1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.
- 10928 2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute  
10929 command on the IAS Zone server’s *IAS\_CIE\_Address* attribute with its IEEE address.
- 10930 3. The IAS Zone server MAY configure a binding table entry for the IAS CIE’s address because all of its  
10931 communication will be directed to the IAS CIE.

- 10932 4. Upon a user input determined by the manufacturer (e.g., a button, change to device’s *ZoneStatus* attribute  
10933 that would result in a Zone Status Change Notification command) and the IAS Zone server’s *ZoneState*  
10934 attribute equal to 0x00 (unenrolled), the IAS Zone server SHALL send a Zone Enroll Request command.  
10935 5. The IAS CIE SHALL send a Zone Enroll Response command, which assigns the IAS Zone server’s  
10936 *ZoneID* attribute.  
10937 6. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

10938 **Auto-Enroll-Response**

- 10939 1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.  
10940 2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute  
10941 command on the IAS Zone server’s *CIE\_IAS\_Address* attribute with its IEEE address.  
10942 3. The IAS Zone server MAY configure a binding table entry for the IAS CIE’s address because all of its  
10943 communication will be directed to the IAS CIE.  
10944 4. The IAS CIE SHALL send a Zone Enroll Response, which assigns the IAS Zone server’s *ZoneID* attribute.  
10945 5. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

10946 **Auto-Enroll-Request**

- 10947 1. After an IAS Zone server is commissioned to a network, the IAS CIE MAY perform service discovery.  
10948 2. If the IAS CIE determines it wants to enroll the IAS Zone server, it SHALL send a Write Attribute  
10949 command on the IAS Zone server’s *IAS\_CIE\_Address* attribute with its IEEE address.  
10950 3. The IAS Zone server MAY configure a binding table entry for the IAS CIE’s address because all of its  
10951 communication will be directed to the IAS CIE.  
10952 4. The IAS Zone server SHALL send a Zone Enroll Request command.  
10953 5. The IAS CIE SHALL send a Zone Enroll Response command, which assigns the IAS Zone server’s  
10954 *ZoneID* attribute.  
10955 6. The IAS Zone server SHALL change its *ZoneState* attribute to 0x01 (enrolled).

10956 Once the *IAS\_CIE\_Address* attribute has been written on an IAS Zone server, the IAS Zone server SHALL only act  
10957 upon commands received from an initiator that matches the *IAS\_CIE\_Address* attribute.

10958 Any attempt via the ZDO bind or unbind request to create, modify or remove binding table entry on a device  
10959 embodying the IAS Zone server SHALL be rejected and responded with the status NOT\_AUTHORIZED, if the  
10960 subjected binding table entry is related to an IAS Zone server cluster and the ZDP request does not come from the  
10961 paired IAS CIE.

10962 **8.2.2.2 Commands Received**

10963 The command IDs received by the IAS Zone server cluster are listed in Table 8-9 Received Command IDs for the IAS  
10964 Zone Cluster.

10965 **Table 8-9. Received Command IDs for the IAS Zone Cluster**

Command Id	Description	M/O
0x00	Zone Enroll Response	M
0x01	Initiate Normal Operation Mode	O
0x02	Initiate Test Mode	O

10966 **8.2.2.2.1 Zone Enroll Response Command**

10967 **8.2.2.2.1.1 Payload Format**

10968 The Zone Enroll Response command payload SHALL be formatted as illustrated in Figure 8-2  
 10969 Format of the Zone Enroll Response Command Payload.

10970 **Figure 8-2. Format of the Zone Enroll Response Command Payload**

<b>Bits</b>	8	8
<b>Data Type</b>	enum8	uint8
<b>Field Name</b>	Enroll response code	Zone ID

10971  
 10972 The permitted values of the Enroll Response Code are shown in Table 8-10  
 10973 Values of the Enroll Response Code.

**Table 8-10. Values of the Enroll Response Code**

Code	Meaning	Details
0x00	Success	Success
0x01	Not supported	This specific Zone type is not known to the CIE and is not supported.
0x02	No enroll permit	CIE does not permit new zones to enroll at this time.
0x03	Too many zones	CIE reached its limit of number of enrolled zones

10974  
 10975 The Zone ID field is the index into the zone table of the CIE (Table 8-12  
 10976 Format of the Zone Table). This field is only relevant if the response code is success.

10977 **8.2.2.2.1.2 Effect on Receipt**

10978 On receipt, the device embodying the Zone server is notified that it is now enrolled as an active alarm device  
 10979 The device embodying the Zone server must authenticate received messages by checking the address of their sender  
 10980 against *IAS\_CIE\_Address*. This is to ensure that only messages from the correct CIE are accepted.

10981 **8.2.2.2.2 Initiate Normal Operation Mode Command**

10982 Used to tell the IAS Zone server to commence normal operation mode.

10983 **8.2.2.2.2.1 Payload Format**

10984 This command has no payload.

10985 **8.2.2.2.2.2 Effect on Receipt**

10986 Upon receipt, the IAS Zone server SHALL commence normal operational mode.  
 10987 Any configurations and changes made (e.g., *CurrentZoneSensitivityLevel* attribute) to the IAS Zone server SHALL  
 10988 be retained.  
 10989 Upon commencing normal operation mode, the IAS Zone server SHALL send a Zone Status Change Notification  
 10990 command updating the *ZoneStatus* attribute Test bit to zero (i.e., “operation mode”).

10991 **8.2.2.2.2.3 Initiate Test Mode Command**

10992 Certain IAS Zone servers MAY have operational configurations that could be configured OTA or locally on the  
 10993 device. This command enables them to be remotely placed into a test mode so that the user or installer MAY configure  
 10994 their field of view, sensitivity, and other operational parameters. They MAY also verify the placement and proper  
 10995 operation of the IAS Zone server, which MAY have been placed in a difficult to reach location (i.e., making a physical  
 10996 input on the device impractical to trigger).

10997 Another use case for this command is large deployments, especially commercial and industrial, where placing the  
 10998 entire IAS system into test mode instead of a single IAS Zone server is infeasible due to the vulnerabilities that might  
 10999 arise. This command enables only a single IAS Zone server to be placed into test mode.

11000 The biggest limitation of this command is that most IAS Zone servers today are battery-powered sleepy nodes that  
 11001 cannot reliably receive commands. However, implementers MAY decide to program an IAS Zone server by factory  
 11002 default to maintain a limited duration of normal polling upon initialization/joining to a new network. Some IAS Zone  
 11003 servers MAY also have AC mains power and are able to receive commands. Some types of IAS Zone servers that  
 11004 MAY benefit from this command are: motion sensors and fire sensor/smoke alarm listeners (i.e., a device that listens  
 11005 for a non-communicating fire sensor to alarm and communicates this to the IAS CIE).

11006 **8.2.2.2.2.4 Payload Format**

11007 The Initiate Test Mode command SHALL be formatted as illustrated below:

11008 **Figure 8-3. Payload format of Initiate Test Mode command**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	Test Mode Duration	Current Zone Sensitivity Level

11009

11010 **8.2.2.2.2.5 Test Mode Duration Field**

11011 Specifies the duration, in seconds, for which the IAS Zone server SHALL operate in its test mode.

11012 **8.2.2.2.2.6 Current Zone Sensitivity Level Field**

11013 Specifies the sensitivity level the IAS Zone server SHALL use for the duration of the Test Mode and with which it  
 11014 must update its *CurrentZoneSensitivityLevel* attribute.

11015 The permitted values of Current Zone Sensitivity Level are shown defined for the *CurrentZoneSensitivityLevel*  
 11016 Attribute in Section 8.2.2.1.2.4.

11017 **8.2.2.2.2.7 Effect on Receipt**

11018 Upon receipt, the IAS Zone server SHALL commence test mode for the duration specified in the command and update  
 11019 its *CurrentZoneSensitivityLevel* attribute to match the value specified in the command.

11020 The IAS Zone server SHALL send a Zone Status Change Notification command updating the *ZoneStatus* attribute  
 11021 Test bit to one (i.e., “test mode”).

11022 While in test mode, the IAS Zone server SHALL send Zone Status Change Notification commands with the  
 11023 appropriate payload to signal that the node is successfully detecting test events.

11024 Upon completing the allotted test mode duration time, the IAS Zone server SHALL resume normal operation mode  
 11025 and SHALL send a Zone Status Change Notification command updating the *ZoneStatus* attribute Test bit to zero (i.e.,  
 11026 “normal operation mode”).

11027 At any time, the IAS Zone client MAY send an Initiate Normal Operation Mode command.

11028 The behavior of the IAS Zone server while in test mode is manufacturer specific. The below devices SHOULD behave  
 11029 in the following manner:

- 11030 • Motion sensor
  - 11031 ○ Suspend battery saving features designed to reduce the frequency of motion detection events sent
  - 11032 to the IAS CIE.
  - 11033 ○ Reduce the “blackout period” (i.e., the period the sensor waits before sending a second motion
  - 11034 detection event) so that the IAS Zone server restores an existing motion event and is ready to send
  - 11035 another motion event notification within ten seconds. This is sometimes known as “walk test”
  - 11036 mode.
  - 11037 ○ Support manufacturer specific profile extensions to allow a user to configure additional
  - 11038 operational parameters that can be tested while in test mode.
- 11039 • Fire sensor
  - 11040 ○ Begin listening for the user to initiate a test mode on the non-communicating fire sensor or smoke
  - 11041 detector
  - 11042 ○ Generate a Zone Status Change Notification command upon detecting an audible test alarm
- 11043 • Carbon monoxide sensor
  - 11044 ○ Begin listening for the user to initiate a test mode on the non-communicating carbon monoxide
  - 11045 detector
  - 11046 ○ Generate a Zone Status Change Notification command upon detecting an audible test alarm

11047 The above guidelines are intended as guidelines for the devices likely to implement test mode. Any IAS Zone server  
 11048 MAY implement test mode features and commands.

11049 Future revisions to this section MAY include additional commands germane to the operational behavior of a given  
 11050 IAS Zone server.

### 11051 8.2.2.3 Commands Generated

11052 The generated command IDs for the IAS Zone server cluster are listed in Table 8-11 Generated Command IDs for the  
 11053 IAS Zone Cluster.

11054 **Table 8-11. Generated Command IDs for the IAS Zone Cluster**

Command Identifier	Description	M/O
0x00	Zone Status Change Notification	M
0x01	Zone Enroll Request	M

#### 11055 8.2.2.3.1 Zone Status Change Notification Command

##### 11056 8.2.2.3.1.1 Payload Format

11057 The Zone Status Change Notification command payload SHALL be formatted as illustrated in Figure 8-4Format of  
11058 the Zone Status Change Notification Command Payload.

11059 **Figure 8-4. Format of the Zone Status Change Notification Command Payload**

<b>Bits</b>	16	8	8	16
<b>Data Type</b>	map16	map8	uint8	uint16
<b>Field Name</b>	Zone Status	Extended Status	Zone ID	Delay

11060 **8.2.2.3.1.2 Zone Status Parameter**

11061 The Zone Status field SHALL be the current value of the *ZoneStatus* attribute.

11062 **8.2.2.3.1.3 Extended Status Parameter**

11063 The Extended Status field is reserved for additional status information and SHALL be set to zero.

11064 **8.2.2.3.1.4 Zone ID Parameter**

11065 Zone ID is the index of the Zone in the CIE's zone table (Table 8-12). If none is programmed, the Zone ID default  
11066 value SHALL be indicated in this field.

11067 **8.2.2.3.1.5 Delay Parameter**

11068 The Delay field is defined as the amount of time, in quarter-seconds, from the moment when a change takes place in  
11069 one or more bits of the Zone Status attribute and the successful transmission of the Zone Status Change Notification.  
11070 This is designed to help congested networks or offline servers quantify the amount of time from when an event was  
11071 detected and when it could be reported to the client.

11072 **8.2.2.3.1.6 When Generated**

11073 The Zone Status Change Notification command is generated when a change takes place in one or more bits of the  
11074 *ZoneStatus* attribute.

11075 **8.2.2.3.2 Zone Enroll Request Command**

11076 **8.2.2.3.2.1 Payload Format**

11077 The Zone Enroll Request command payload SHALL be formatted as illustrated in Figure 8-5Format of the Zone  
11078 Enroll Request Command Payload.

11079 **Figure 8-5. Format of the Zone Enroll Request Command Payload**

<b>Bits</b>	16	16
<b>Data Type</b>	enum16	uint16
<b>Field Name</b>	Zone Type	Manufacturer Code

11080  
11081 The Zone Type field SHALL be the current value of the *ZoneType* attribute.

11082 The Manufacturer Code field SHALL be the manufacturer code as held in the node descriptor for the device. See  
11083 [Z12] Manufacturer Code Database.

11084 **8.2.2.3.2.2 When Generated**

11085 The Zone Enroll Request command is generated when a device embodying the Zone server cluster wishes to be  
 11086 enrolled as an active alarm device. It must do this immediately it has joined the network (during commissioning).

11087 **8.2.3 Client**

11088 No dependencies or cluster specific attributes are defined for the client. The client receives the cluster specific  
 11089 commands detailed in 8.2.2.3. The client generates the cluster specific commands detailed in 8.2.2.2, as required by  
 11090 the application.

11091 **8.3 IAS ACE**

11092 **8.3.1 Overview**

11093 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 11094 etc.

11095 The IAS ACE cluster defines an interface to the functionality of any Ancillary Control Equipment of the IAS system.  
 11096 Using this cluster, an ACE device can access a IAS CIE device and manipulate the IAS system, on behalf of a level-  
 11097 2 user (see [N1]).

11098 The client is usually implemented by the IAS ACE device. It allows the IAS ACE device to control the IAS CIE  
 11099 device, which typically implements the server side.

11100 **8.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; ZHA 1.2 and 1.2.1 features; CCB 1977

11101 **8.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASACE	Type 1 (client to server)

11102 **8.3.1.3 Cluster Identifiers**

Identifier	Name
0x0501	IAS ACE

11103 **8.3.2 Server**

11104 **8.3.2.1 Attributes**

11105 No attributes are currently defined for this cluster.

11106 **8.3.2.2 Zone Table**

11107 The Zone Table is used to store information for each Zone enrolled by the CIE. The maximum number of entries in  
11108 the table is 255.

11109 The format of a group table entry is illustrated in Table 8-12.

11110 **Table 8-12. Format of the Zone Table**

Field	Type	Valid Range	Description
Zone ID	uint8	0x00 – 0xfe	The unique identifier of the zone
Zone Type	enum16	0x0000 – 0xffff	See Table 8-5.
Zone Address	EUI64	Valid 64-bit IEEE address	Device address

11111 The Zone ID is a unique reference number allocated by the CIE at zone enrollment time.

11112 The Zone ID is used by IAS devices to reference specific zones when communicating with the CIE. The Zone ID of  
11113 each zone stays fixed until that zone is un-enrolled.

11114 **8.3.2.3 Commands Received**

11115 The received command IDs for the IAS ACE server cluster are listed in Table 8-13  
11116 Received Command IDs for the IAS ACE Cluster.

11117 **Table 8-13. Received Command IDs for the IAS ACE Cluster**

Command Identifier	Description	M/O
0x00	Arm	M
0x01	Bypass	M
0x02	Emergency	M
0x03	Fire	M
0x04	Panic	M
0x05	Get Zone ID Map	M
0x06	Get Zone Information	M
0x07	Get Panel Status	M
0x08	Get Bypassed Zone List	M
0x09	Get Zone Status	M



11118 **8.3.2.3.1 Arm Command**

11119 **8.3.2.3.1.1 Payload Format**

11120 The Arm command payload SHALL be formatted as illustrated in Figure 8-6.

11121 **Figure 8-6. Format of the Arm Command Payload**

<b>Bits</b>	8	Varies	8
<b>Data Type</b>	enum8	string	uint8
<b>Field Name</b>	Arm Mode	Arm/Disarm Code	Zone ID

11122 **8.3.2.3.1.2 Arm Mode Field**

11123 The Arm Mode field SHALL have one of the values shown in Table 8-14 Arm Mode Field Values.

11124 **Table 8-14. Arm Mode Field Values**

<b>Value</b>	<b>Meaning</b>
0x00	Disarm
0x01	Arm Day/Home Zones Only
0x02	Arm Night/Sleep Zones Only
0x03	Arm All Zones

11125 **8.3.2.3.1.3 Arm/Disarm Code Field**

11126 The Arm/Disarm Code SHALL be a code entered into the ACE client (e.g., security keypad) or system by the user  
 11127 upon arming/disarming. The server MAY validate the Arm/Disarm Code received from the IAS ACE client in Arm  
 11128 command payload before arming or disarming the system. If the client does not have the capability to input an  
 11129 Arm/Disarm Code (e.g., keyfob), or the system does not require one, the client SHALL a transmit a string with a  
 11130 length of zero.

11131 There is no minimum or maximum length to the Arm/Disarm Code; however, the Arm/Disarm Code  
 11132 SHOULD be between four and eight alphanumeric characters in length.

11133 The string encoding SHALL be UTF-8.

11134 **8.3.2.3.1.4 Zone ID Field**

11135 Zone ID is the index of the Zone in the CIE's zone table (Table 8-12). If none is programmed, the Zone ID default  
 11136 value SHALL be indicated in this field.

11137 **8.3.2.3.1.5 Effect on Receipt**

11138 On receipt of this command, the receiving device sets its arm mode according to the value of the Arm Mode field, as  
 11139 detailed in Table 8-14. It is not guaranteed that an Arm command will succeed. Based on the current state of the IAS  
 11140 CIE, and its related devices, the command can be rejected. The device SHALL generate an Arm Response command  
 11141 (see 8.3.2.4.1) to indicate the resulting armed state.

**11142 8.3.2.3.2 Bypass Command**

11143 Provides IAS ACE clients with a method to send zone bypass requests to the IAS ACE server. Bypassed zones MAY  
11144 be faulted or in alarm but will not trigger the security system to go into alarm. For example, a user MAY wish to  
11145 allow certain windows in his premises protected by an IAS Zone server to be left open while the user leaves the  
11146 premises. The user could bypass the IAS Zone server protecting the window on his IAS ACE client (e.g., security  
11147 keypad), and if the IAS ACE server indicates that zone is successfully bypassed, arm his security system while he is  
11148 away.

**11149 8.3.2.3.2.1 Payload Format**

11150 The Bypass command payload SHALL be formatted as illustrated in Figure 8-7For.

11151 **Figure 8-7. Format of the Bypass Command Payload**

<b>Bits</b>	8	8	...	8	Varies
<b>Data Type</b>	uint8	uint8	...	uint8	string
<b>Field Name</b>	Number of Zones	Zone ID	...	Zone ID	Arm/Disarm Code

**11152 8.3.2.3.2.2 Number of Zones Field**

11153 This is the number of Zone IDs included in the payload.

**11154 8.3.2.3.2.3 Zone ID Field**

11155 Zone ID is the index of the Zone in the CIE's zone table (Table 8-12Format of the Zone Table).

**11156 8.3.2.3.2.4 Arm/Disarm Code Field**

11157 This field is the same as the Arm/Disarm Code field defined in Section 8.3.2.3.1.3.

**11158 8.3.2.3.2.5 Effect of Receipt**

11159 On receipt of this command, the IAS ACE server SHALL process this bypass request and generate a single Bypass  
11160 Response command for the zones requested in the Bypass command payload

**11161 8.3.2.3.3 Emergency, Fire and Panic Commands**

11162 These commands indicate the emergency situations inherent in their names. They have no payload.

**11163 8.3.2.3.4 Get Zone ID Map Command****11164 8.3.2.3.4.1 Payload Format**

11165 This command has no payload.

**11166 8.3.2.3.4.2 Effect on Receipt**

11167 On receipt of this command, the device SHALL generate a Get Zone ID Map Response command. See 8.3.2.4.2.

**11168 8.3.2.3.5 Get Zone Information Command****11169 8.3.2.3.5.1 Payload Format**

11170 The Get Zone Information command payload SHALL be formatted as illustrated in Figure 8-8.

11171

**Figure 8-8. Format of the Get Zone Information Command Payload**

<b>Bits</b>	8
<b>Data Type</b>	uint8
<b>Field Name</b>	Zone ID

11172 **8.3.2.3.5.2 Effect on Receipt**

11173 On receipt of this command, the device SHALL generate a Get Zone Information Response command. See 8.3.2.4.3.

11174 **8.3.2.3.6 Get Panel Status Command**

11175 This command is used by ACE clients to request an update to the status (e.g., security system arm state) of the ACE  
 11176 server (i.e., the IAS CIE). This command is useful for battery-powered ACE clients with polling rates longer than the  
 11177 standard check-in rate.

11178 **8.3.2.3.6.1 Payload Format**

11179 There is no payload for the Get Panel Status command.

11180 **8.3.2.3.6.2 Effect on Receipt**

11181 On receipt of this command, the ACE server responds with the status of the security system. The IAS ACE server  
 11182 SHALL generate a Get Panel Status Response command.

11183 **8.3.2.3.7 Get Bypassed Zone List Command**

11184 Provides IAS ACE clients with a way to retrieve the list of zones to be bypassed. This provides them with the ability  
 11185 to provide greater local functionality (i.e., at the IAS ACE client) for users to modify the Bypassed Zone List and  
 11186 reduce communications to the IAS ACE server when trying to arm the CIE security system.

11187 **8.3.2.3.7.1 Payload Format**

11188 This command has no payload.

11189 **8.3.2.3.7.2 Effect on Receipt**

11190 Upon receipt, the IAS ACE server sends a Set Bypassed Zone List command.

11191 **8.3.2.3.8 Get Zone Status Command**

11192 This command is used by ACE clients to request an update of the status of the IAS Zone devices managed by the ACE  
 11193 server (i.e., the IAS CIE). This command is useful for battery-powered ACE clients with polling rates longer than the  
 11194 standard check-in rate. The command is similar to the Get Attributes Supported command in that it specifies a starting  
 11195 Zone ID and a number of Zone IDs for which information is requested.

11196 Depending on the number of IAS Zone devices managed by the IAS ACE server, sending the Zone Status of all zones  
 11197 MAY not fit into a single Get Zone Status Response command. IAS ACE clients MAY need to send multiple Get  
 11198 Zone Status commands in order to get the information they seek.

11199 **8.3.2.3.8.1 Payload Format**

11200 The Get Zone Status command SHALL be formatted as illustrated below.

11201

**Figure 8-9. Format of the Get Zone Status command**

<b>Bits</b>	8	8	8	16
<b>Data Type</b>	uint8	uint8	bool	map16
<b>Field Name</b>	Starting Zone ID	Max Number of Zone IDs	Zone Status Mask Flag	Zone Status Mask

11202

**11203 8.3.2.3.8.2 Starting Zone ID Field**

11204 Specifies the starting Zone ID at which the IAS Client would like to obtain zone status information.

**11205 8.3.2.3.8.3 Max Number of Zone IDs Requested Field**

11206 Specifies the maximum number of Zone IDs and corresponding Zone Statuses that are to be returned by the IAS ACE  
11207 server when it responds with a Get Zone Status Response command.

**11208 8.3.2.3.8.4 Zone Status Mask Flag Field**

11209 Functions as a query operand with the Zone Status Mask field. If set to zero (i.e., FALSE), the IAS ACE server  
11210 SHALL include all Zone IDs and their status, regardless of their Zone Status when it responds with a Get Zone Status  
11211 Response command.

11212 If set to one (i.e., TRUE), the IAS ACE server SHALL include only those Zone IDs whose *Zone Status* attribute is  
11213 equal to one or more of the Zone Statuses requested in the Zone Status Mask field of the Get Zone Status command.

11214 Use of Zone Status Mask Flag and Zone Status Mask fields allow a client to obtain updated information for the subset  
11215 of Zone IDs they're interested in, which is beneficial when the number of IAS Zone devices in a system is large.

**11216 8.3.2.3.8.5 Zone Status Mask Field**

11217 Coupled with the Zone Status Mask Flag field, functions as a mask to enable IAS ACE clients to get information about  
11218 the Zone IDs whose *ZoneStatus* attribute is equal to any of the bits indicated by the IAS ACE client in the Zone Status  
11219 Mask field. The format of this field is the same as the *ZoneStatus* attribute in the IAS Zone cluster. Per the Zone  
11220 Status Mask Flag field, IAS ACE servers SHALL respond with only the Zone IDs whose *ZoneStatus* attributes are  
11221 equal to at least one of the Zone Status bits set in the Zone Status Mask field requested by the IAS ACE client.

11222 For example, if the Zone Status Mask field set to "0x0003" would match IAS Zones whose *ZoneStatus* attributes are  
11223 0x0001, 0x0002, and 0x0003. In other words, if a logical 'AND' between the Zone Status Mask field and the IAS  
11224 Zone's *ZoneStatus* attribute yields a non-zero result, the IAS ACE server SHALL include that IAS Zone in the Get  
11225 Zone Status Response command.

**11226 8.3.2.3.8.6 Effect on Receipt**

11227 On receipt of this command, the IAS ACE server responds with the status of the zones it manages that meet those  
11228 requested in the Get Zone Status command. The IAS ACE server SHALL generate a Get Zone Status Response  
11229 command.

**11230 8.3.2.4 Commands Generated**

11231 The generated command IDs for the IAS ACE server cluster are listed in Table 8-15.

11232

**Table 8-15. Generated Command IDs for the IAS ACE Cluster**

Command Identifier	Description	M/O
0x00	Arm Response	M
0x01	Get Zone ID Map Response	M
0x02	Get Zone Information Response	M
0x03	Zone Status Changed	M
0x04	Panel Status Changed	M
0x05	Get Panel Status Response	M
0x06	Set Bypassed Zone List	M
0x07	Bypass Response	M
0x08	Get Zone Status Response	M

11233 **8.3.2.4.1 Arm Response Command**

11234 **8.3.2.4.1.1 Payload Format**

11235 The Arm Response command payload SHALL be formatted as illustrated in Figure 8-10.

11236 **Figure 8-10. Format of the Arm Response Command Payload**

<b>Bits</b>	8
<b>Data Type</b>	enum8
<b>Field Name</b>	Arm Notification

11237 **8.3.2.4.1.2 Arm Notification Field**

11238 The Arm Notification field SHALL have one of the values shown in Table 8-16.

11239 **Table 8-16. Arm Notification Values**

Value	Meaning
0x00	All Zones Disarmed
0x01	Only Day/Home Zones Armed
0x02	Only Night/Sleep Zones Armed
0x03	All Zones Armed
0x04	Invalid Arm/Disarm Code

Value	Meaning
0x05	Not ready to arm*
0x06	Already disarmed

11240 \* NOTE: reasons for not being ready to arm are determined by the IAS ACE server manufacturer

### 11241 8.3.2.4.2 Get Zone ID Map Response Command

#### 11242 8.3.2.4.2.1 Payload Format

11243 The Get Zone ID Map Response command payload SHALL be formatted as illustrated in Figure 8-11.

11244 **Figure 8-11. Get Zone ID Map Response Command Payload**

<b>Bits</b>	16	...	16
<b>Data Type</b>	map16	...	map16
<b>Field Name</b>	Zone ID Map section 0	...	Zone ID Map section 15

11245  
11246 The 16 fields of the payload indicate whether each of the Zone IDs from 0 to 0xff is allocated or not. If bit n of Zone  
11247 ID Map section N is set to 1, then Zone ID (16 x N + n ) is allocated, else it is not allocated.

### 11248 8.3.2.4.3 Get Zone Information Response Command

#### 11249 8.3.2.4.3.1 Payload Format

11250 The Get Zone Information Response command payload SHALL be formatted as illustrated in Figure 8-12.

11251 **Figure 8-12. Format of the Get Zone Information Response Command Payload**

<b>Bits</b>	8	16	64	Varies
<b>Data Type</b>	uint8	enum16	EUI64	string
<b>Field Name</b>	Zone ID	Zone Type	IEEE address	Zone Label

11252  
11253 The first 3 fields of the payload are equal to the fields of the Zone Table entry corresponding to the ZoneID field of  
11254 the Get Zone Information command to which this command is a response.

11255 If the Zone ID is unallocated, this SHALL be indicated by setting the Zone Type and IEEE Address fields to 0xffff  
11256 (see Table 8-5 Values of the ) and 0xffffffffffffffff respectively.

#### 11257 8.3.2.4.3.2 Zone Label Field

11258 Provides the Zone Label stored in the IAS CIE. If none is programmed, the IAS ACE server SHALL transmit a string  
11259 with a length of zero.

11260 There is no minimum or maximum length to the Zone Label field; however, the Zone Label SHOULD be between 16  
11261 to 24 alphanumeric characters in length.

11262 The string encoding SHALL be UTF-8.

11263 **8.3.2.4.4 Zone Status Changed Command**

11264 This command updates ACE clients in the system of changes to zone status recorded by the ACE server (e.g., IAS  
 11265 CIE device).

11266 An IAS ACE server SHOULD send a Zone Status Changed command upon a change to an IAS Zone device’s  
 11267 *ZoneStatus* that it manages (i.e., IAS ACE server SHOULD send a Zone Status Changed command upon receipt of a  
 11268 Zone Status Change Notification command).

11269 **8.3.2.4.4.1 Payload Format**

11270 The Zone Status Changed Command SHALL be formatted as illustrated in Figure 8-13.

11271 **Figure 8-13. Format of the Zone Status Changed Command Payload**

<b>Bits</b>	8	16	8	Varies
<b>Data Type</b>	uint8	enum16	enum8	string
<b>Field Name</b>	Zone ID	Zone Status	Audible Notification	Zone Label

11272 **8.3.2.4.4.2 Zone ID Field**

11273 The index of the Zone in the CIE’s zone table (Table 8-12). If none is programmed, the ZoneID attribute default value  
 11274 SHALL be indicated in this field.

11275 **8.3.2.4.4.3 Zone Status Field**

11276 The current value of the ZoneStatus attribute.

11277 **8.3.2.4.4.4 Audible Notification Field**

11278 Provide the ACE client with information on which type of audible notification it SHOULD make for the zone status  
 11279 change. This field is useful for telling the ACE client to play a standard chime or other audio indication or to mute  
 11280 and not sound an audible notification at all. This field also allows manufacturers to create additional audible alert  
 11281 types (e.g., dog barking, windchimes, conga drums) to enable users to customize their system.

11282 The Audible Notification field SHALL be formatted as illustrated below:

11283 **Figure 8-14. Audible Notification field value**

<b>Enumeration</b>	<b>Description</b>
0x00	Mute (i.e., no audible notification)
0x01	Default sound
0x80-0xff	Manufacturer specific

11284 The default value SHALL be 0x01.

11285 **8.3.2.4.4.5 Zone Label Field**

11286 Provides the Zone Label stored in the IAS CIE. If none is programmed, the IAS ACE server SHALL transmit a string  
 11287 with a length of zero.

11288 There is no minimum or maximum length to the Zone Label field; however, the Zone Label SHOULD be between 16  
11289 to 24 alphanumeric characters in length.

11290 The string encoding SHALL be UTF-8.

### 11291 **8.3.2.4.5 Panel Status Changed Command**

11292 This command updates ACE clients in the system of changes to panel status recorded by the ACE server (e.g., IAS  
11293 CIE device).

11294 Sending the Panel Status Changed command (vs. the Get Panel Status and Get Panel Status Response method) is  
11295 generally useful only when there are IAS ACE clients that data poll within the retry timeout of the network (e.g., less  
11296 than 7.68 seconds).

11297 An IAS ACE server SHALL send a Panel Status Changed command upon a change to the IAS CIE’s panel status  
11298 (e.g., Disarmed to Arming Away/Stay/Night, Arming Away/Stay/Night to Armed, Armed to Disarmed) as defined in  
11299 the Panel Status field.

11300 When Panel Status is Arming Away/Stay/Night, an IAS ACE server SHOULD send Panel Status Changed commands  
11301 every second in order to update the Seconds Remaining. In some markets (e.g., North America), the final 10 seconds  
11302 of the Arming Away/Stay/Night sequence requires a separate audible notification (e.g., a double tone).

#### 11303 **8.3.2.4.5.1 Payload Format**

11304 The Panel Status Changed Command SHALL be formatted as illustrated in Figure 8-15.

11305 **Figure 8-15. Format of the Panel Status Changed Command Payload**

Bits	8	8	8	8
Data Type	enum8	uint8	enum8	enum8
Field Name	Panel Status	Seconds Remaining	Audible Notification	Alarm Status

#### 11306 **8.3.2.4.5.2 PanelStatus Parameter**

11307 The Panel Status parameter SHALL be formatted as illustrated in Table 8-17.

11308 **Table 8-17. PanelStatus Field Values**

Panel Status Enumerations	Description
0x00	Panel disarmed (all zones disarmed) and ready to arm
0x01	Armed stay
0x02	Armed night
0x03	Armed away
0x04	Exit delay
0x05	Entry delay
0x06	Not ready to arm



Panel Status Enumerations	Description
0x07	In alarm
0x08	Arming Stay
0x09	Arming Night
0x0a	Arming Away

11309 **8.3.2.4.5.3 Audible Notification Field**

11310 See 8.3.2.4.4.4 for a description of this field.

11311 **8.3.2.4.5.4 Alarm Status Field**

11312 Provides the ACE client with information on the type of alarm the panel is in if its Panel Status field indicates it is “in  
 11313 alarm.” This field MAY be useful for ACE clients to display or otherwise initiate notification for users.

11314 The Alarm Status field SHALL be formatted as illustrated below. Note: this is the same as the Warning Mode field  
 11315 in the IAS WD cluster.

11316 **Figure 8-16. Alarm Status field value**

Enumeration	Description
0x00	No alarm
0x01	Burglar
0x02	Fire
0x03	Emergency
0x04	Police Panic
0x05	Fire Panic
0x06	Emergency Panic (i.e., medical issue)

11317

11318 The default value SHALL be 0x00.

11319 **8.3.2.4.5.5 Seconds Remaining Parameter**

11320 Indicates the number of seconds remaining for the server to be in the state indicated in the Panel Status parameter.

11321 The Seconds Remaining parameter SHALL be provided if the Panel Status parameter has a value of 0x04 (Exit delay)  
 11322 or 0x05 (Entry delay).

11323 The default value SHALL be 0x00.

11324 **8.3.2.4.6 Get Panel Status Response Command**

11325 This command updates requesting IAS ACE clients in the system of changes to the security panel status recorded by  
11326 the ACE server (e.g., IAS CIE device).

11327 **8.3.2.4.6.1 Payload Format**

11328 The Get Panel Status Response command SHALL be formatted as illustrated below.

11329 **Figure 8-17. Get Panel Status Response command**

<b>Bits</b>	8	8	8	8
<b>Data Type</b>	enum8	uint8	enum8	enum8
<b>Field Name</b>	Panel Status	Seconds Remaining	Audible Notification	Alarm Status

11330

11331 **8.3.2.4.6.2 Panel Status Field**

11332 See 8.3.2.4.5.2 for a description of this field.

11333 **8.3.2.4.6.3 Seconds Remaining Field**

11334 Indicates the number of seconds remaining for the server to be in the state indicated in the Panel Status field. The  
11335 Seconds Remaining field SHALL be provided if the Panel Status field has a value of 0x04 (Exit delay) or 0x05 (Entry  
11336 delay).

11337 The default value SHALL be 0x00.

11338 **8.3.2.4.6.4 Audible Notification Field**

11339 See 8.3.2.4.4.4 for a description of this field.

11340 **8.3.2.4.6.5 Alarm Status Field**

11341 See 8.3.2.4.5.4 for a description of this field.

11342 **8.3.2.4.7 Set Bypassed Zone List Command**

11343 Sets the list of bypassed zones on the IAS ACE client. This command can be sent either as a response to the Get  
11344 Bypassed Zone List command or unsolicited when the list of bypassed zones changes on the ACE server.

11345 **8.3.2.4.7.1 Payload Format**

11346 The Set Bypassed Zone List command SHALL be formatted as illustrated below.

11347 **Figure 8-18. Set Bypassed Zone List Command payload format**

<b>Bits</b>	8	8	8	...	8
<b>Data Type</b>	uint8	uint8	uint8	...	uint8

<b>Field Name</b>	Number of Zones	Zone ID 1	Zone ID 2	...	Zone ID <i>n</i>
-------------------	-----------------	-----------	-----------	-----	------------------

11348

11349 **8.3.2.4.7.2 Number of Zones Field**

11350 This is the number of Zone IDs included in the payload.

11351 If no zones are bypassed, the IAS ACE server SHALL send the Set Bypassed Zone List command with a Number of  
 11352 Zones field set to “0” (zero).

11353 **8.3.2.4.7.3 Zone ID 1...X Field**

11354 Zone ID is the index of the Zone in the CIE's zone table and is an array of Zone IDs for each zone that is bypassed  
 11355 where X is equal to the value of the Number of Zones field. There is no order imposed by the numbering of the Zone  
 11356 ID field in this command payload. IAS ACE servers SHOULD provide the array of Zone IDs in ascending order.

11357 **8.3.2.4.7.4 Implementation Guidelines**

11358 The IAS ACE server SHALL reset (i.e., set to be “not bypassed”) all previously bypassed zones each time the security  
 11359 system is disarmed unless certain zones are programmed by the user to be bypassed on a permanent basis.

11360 **8.3.2.4.8 Bypass Response Command**

11361 Provides the response of the security panel to the request from the IAS ACE client to bypass zones via a Bypass  
 11362 command.

11363 **8.3.2.4.8.1 Payload Format**

11364 The Bypass Response command SHALL be formatted as illustrated below:

11365 **Figure 8-19. Bypass Response command format**

<b>Bits</b>	8	8	8	...	8
<b>Data Type</b>	uint8	uint8	uint8	...	uint8
<b>Field Name</b>	Number of Zones	Bypass Result for Zone ID 1	Bypass Result for Zone ID 2	...	Bypass Result for Zone ID <i>n</i>

11366

11367 **8.3.2.4.8.2 Number of Zones Field**

11368 This is the number of Zone IDs for which a bypass result is provided in the payload.

11369 **8.3.2.4.8.3 Bypass Result Field**

11370 An array of Zone IDs for each zone requested to be bypassed via the Bypass command where X is equal to the value  
 11371 of the Number of Zones field. The order of results for Zone IDs SHALL be the same as the order of Zone IDs sent in  
 11372 the Bypass command by the IAS ACE client.

11373 The permitted values of Bypass Result are shown below:

11374

**Table 8-18. Values of Bypass Result Field**

Value	Meaning	Description
0x00	Zone bypassed	The Zone ID requested to be bypassed is successful. Zone is bypassed.
0x01	Zone not bypassed	The Zone ID requested to be bypassed is unsuccessful. Zone is not bypassed.
0x02	Not allowed	The Zone ID requested to be bypassed is not eligible to be bypassed per the policy or user configurations on the IAS ACE server. Zone is not bypassed.
0x03	Invalid Zone ID	The Zone ID requested to be bypassed is not in the valid range of Zone IDs.
0x04	Unknown Zone ID	The Zone ID requested to be bypassed is in the valid range of Zone IDs, but the IAS ACE server does not have a record of the Zone ID requested.
0x05	Invalid Arm/Disarm Code	A value returned indicating that the Arm/Disarm Code was entered incorrectly.

11375 **8.3.2.4.9 Get Zone Status Response Command**

11376 This command updates requesting IAS ACE clients in the system of changes to the IAS Zone server statuses recorded  
11377 by the ACE server (e.g., IAS CIE device).

11378 **8.3.2.4.9.1 Payload Format**

11379 The Get Zone Status Response command SHALL be formatted as illustrated below.

11380 **Figure 8-20. Format of the Get Zone Status Response command**

<b>Bits</b>	8	8	8	16
<b>Data Type</b>	Boolean	uint8	uint8	map16
<b>Field Name</b>	Zone Status Complete	Number of Zones	Zone ID 1	Zone ID 1 Zone Status

11381

<b>Bits</b>	8	16	...	8	16
<b>Data Type</b>	uint8	map16	...	uint8	map16
<b>Field Name</b>	Zone ID 2	Zone ID 2 Zone Status	...	Zone ID N	Zone ID n Zone Status

11382 **8.3.2.4.9.2 Zone Status Complete Field**

11383 Indicates whether there are additional Zone IDs managed by the IAS ACE Server with Zone Status information to be  
 11384 obtained. A value of zero (i.e., FALSE) indicates there are additional Zone IDs for which Zone Status information is  
 11385 available and that the IAS ACE client SHOULD send another Get Zone Status command.

11386 A value of one (i.e., TRUE) indicates there are no more Zone IDs for the IAS ACE client to query and the IAS ACE  
 11387 client has received all the Zone Status information for all IAS Zones managed by the IAS ACE server. The IAS ACE  
 11388 client SHOULD NOT typically send another Get Zone Status command.

11389 **8.3.2.4.9.3 Number of Zones Field**

11390 This is the number of Zone IDs for which a zone status result is provided in the payload.

11391 **8.3.2.4.9.4 Zone ID Field**

11392 The index of the Zone in the CIE’s zone table. If none is programmed, the *ZoneID* attribute default value SHALL be  
 11393 indicated in this field.

11394 **8.3.2.4.9.5 Zone Status Field**

11395 The current value of the *ZoneStatus* attribute for the indicated Zone ID.

11396 **8.3.3 Client**

11397 The client supports no cluster specific attributes. The client receives the cluster specific commands detailed in  
 11398 8.3.2.4. The client cluster generates the commands detailed in 8.3.2.3, as required by the application.

11399 **8.4 IAS WD**

---

11400 **8.4.1 Overview**

11401 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 11402 etc.

11403 The IAS WD cluster provides an interface to the functionality of any Warning Device equipment of the IAS system.  
 11404 Using this cluster, a CIE device can access an IAS WD device and issue alarm warning indications (siren, strobe  
 11405 lighting, etc.) when a system alarm condition is detected (according to [N1]).

11406 **8.4.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	CCB 2350 2341

11407 **8.4.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	IASWD	Type 1 (client to server)

11408 **8.4.1.3 Cluster Identifiers**

Identifier	Name
0x0502	IAS WD

11409 **8.4.2 Server**11410 **8.4.2.1 Attributes**

11411 The attributes defined for the server cluster are detailed in Table 8-19.

11412 **Table 8-19. Attributes of the IAS WD (Server) Cluster**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>MaxDuration</i>	uint16	0x0000 – 0xffff	RW	240	M

11413 **8.4.2.1.1 *MaxDuration* Attribute**11414 The *MaxDuration* attribute specifies the maximum time in seconds that the siren will sound continuously, regardless  
11415 of start/stop commands.11416 **8.4.2.2 Commands Received**

11417 The received command IDs are listed in Table 8-20.

11418 **Table 8-20. Received Command IDs for the IAS WD Server Cluster**

Command Identifier	Description	M/O
0x00	Start warning	M
0x01	Squawk	M

11419 **8.4.2.2.1 Start Warning Command**11420 This command starts the WD operation. The WD alerts the surrounding area by audible (siren) and visual (strobe)  
11421 signals.11422 A Start Warning command SHALL always terminate the effect of any previous IAS WD cluster command that is still  
11423 current.11424 **8.4.2.2.1.1 Payload Format**

11425 The Start Warning command payload SHALL be formatted as illustrated in Figure 8-21.

11426

**Figure 8-21. Format of the Start Siren Command Payload**

<b>Bits</b>	4	2	2	16	8	8
<b>Data Type</b>	map8			uint16	uint8	enum8
<b>Field Name</b>	Warning Mode	Strobe	Siren Level	Warning Duration	Strobe Duty Cycle	Strobe Level

11427

11428 The Warning Mode and Strobe subfields are concatenated together to a single 8-bit bitmap field. The groups of bits  
 11429 these subfields occupy are used as described below.

11430 **8.4.2.2.1.2 Warning Mode Field**

11431 The Warning Mode field is used as an 4-bit enumeration, can have one of the values defined below. The exact behavior  
 11432 of the WD device in each mode is according to the relevant security standards.

11433 **Table 8-21. Warning Modes**

Warning Mode	Meaning
0	Stop (no warning)
1	Burglar
2	Fire
3	Emergency
4	Police panic
5	Fire panic
6	Emergency Panic (i.e., medical issue)

11434 **8.4.2.2.1.3 Strobe Field**

11435 The Strobe field is used as a 2-bit enumeration, and determines if the visual indication is required in addition to the  
 11436 audible siren, as indicated in Table 8-22. If the strobe field is “1” and the Warning Mode is “0” (“Stop”) then only the  
 11437 strobe is activated.

11438 **Table 8-22. Values of the Strobe Field**

Value	Meaning
0	No strobe
1	Use strobe in parallel to warning

11439 **8.4.2.2.1.4 Siren Level Field**

11440 The Siren Level field is used as a 2-bit enumeration, and indicates the intensity of audible squawk sound as shown in  
11441 the following table. At least one level of sound SHALL be supported<sup>121</sup>.

11442 **Table 8-23. Siren Level Field Values**

<i>SirenLevel</i> Value	Description
0	Low level sound
1	Medium level sound
2	High level sound
3	Very high level sound

11443 **8.4.2.2.1.5 Warning Duration Field**

11444 Requested duration of warning, in seconds. If both Strobe and Warning Mode are "0" this field SHALL be ignored.

11445 **8.4.2.2.1.6 Strobe Duty Cycle Field**

11446 Indicates the length of the flash cycle. This provides a means of varying the flash duration for different alarm types  
11447 (e.g., fire, police, burglar). Valid range is 0-100 in increments of 10. All other values SHALL be rounded to the nearest  
11448 valid value. Strobe SHALL calculate duty cycle over a duration of one second. The ON state SHALL precede the OFF  
11449 state. For example, if Strobe Duty Cycle Field specifies "40," then the strobe SHALL flash ON for 4/10ths of a second  
11450 and then turn OFF for 6/10ths of a second.

11451 The default value for this field SHALL be 0x00.

11452 **8.4.2.2.1.7 Strobe Level Field**

11453 Indicates the intensity of the strobe as shown in the table below. This attribute is designed to vary the output of the  
11454 strobe (i.e., brightness) and not its frequency, which is detailed in 8.4.2.2.1.6. At least one level of strobe SHALL be  
11455 supported<sup>122</sup>.

11456 **Table 8-24. Strobe Level Field Values**

<i>StrobeLevel</i> Enumerations	Description
0x00	Low level strobe
0x01	Medium level strobe
0x02	High level strobe
0x03	Very high level strobe

11457 **8.4.2.2.2 Squawk Command**

11458 This command uses the WD capabilities to emit a quick audible/visible pulse called a "squawk". The squawk command  
11459 has no effect if the WD is currently active (warning in progress).

11460 **8.4.2.2.2.1 Payload Format**

<sup>121</sup> CCB 2350 2341 clarify that only one level need be supported

<sup>122</sup> CCB 2350 2341 clarify that only one level need be supported



11461 The Squawk command payload SHALL be formatted as illustrated in Figure 8-22.

11462 **Figure 8-22. Format of the Start Siren Command Payload**

<b>Bits</b>	4	1	1	2
<b>Data Type</b>	map8			
<b>Field Name</b>	Squawk mode	Strobe	Reserved	Squawk level

11463 **8.4.2.2.2.2 Squawk Mode Field**

11464 The Squawk Mode field is used as a 4-bit enumeration, and can have one of the values shown in Table 8-25 Squawk  
 11465 Mode Field. The exact operation of each mode (how the WD “squawks”) is implementation specific.

11466 **Table 8-25. Squawk Mode Field**

Warning Mode	Meaning
0	Notification sound for “System is armed”
1	Notification sound for "System is disarmed"

11467 **8.4.2.2.2.3 Strobe Field**

11468 The strobe field is used as a Boolean, and determines if the visual indication is also required in addition to the audible  
 11469 squawk, as shown in Table 8-26 Strobe Bit.

11470 **Table 8-26. Strobe Bit**

Value	Meaning
0	No strobe
1	Use strobe blink in parallel to squawk

11471 **8.4.2.2.2.4 Squawk Level Field**

11472 The squawk level field is used as a 2-bit enumeration, and determines the intensity of audible squawk sound as shown  
 11473 in Table 8-27 Squawk Level Field Values.

11474 **Table 8-27. Squawk Level Field Values**

Value	Meaning
0	Low level sound
1	Medium level sound
2	High level sound
3	Very High level sound

11475 **8.4.2.3 Commands Generated**

11476 No cluster specific commands are generated by the server cluster.

11477 **8.4.3 Client**

11478 The client side is implemented by the CIE. The CIE is a client of the warning service provided by this cluster. Usually  
11479 a WD would implement an IAS WD cluster server and an IAS Zone cluster server.

11480 There are no cluster specific attributes defined for the client cluster. The client receives no cluster specific commands.

11481 The client cluster generates the cluster specific commands detailed in 8.4.2.2, as required by the application.

# CHAPTER 9 PROTOCOL INTERFACES

11482  
 11483 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 11484 a list of all chapters and documents. References between chapters are made using a X.Y notation where X is the chapter  
 11485 and Y is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 11486 using [Rn] notation.

## 9.1 General Description

### 9.1.1 Introduction

11487  
 11488  
 11489 The clusters specified in this document are for use in applications which interface to external protocols.

### 9.1.2 Cluster List

11490  
 11491 This section lists the clusters specified in this document, and gives examples of typical usage for the purpose of  
 11492 clarification.

11493 The clusters defined in this document are listed in Table 9-1.

11494 **Table 9-1. Clusters of the Protocol Interfaces Functional**

Cluster ID	Cluster Name	Description
0x0016	Partition	The commands and attributes for enabling partitioning of a large frame between devices
0x0600	Generic tunnel	The minimum common commands and attributes required to tunnel any protocol.
0x0601	BACnet protocol tunnel	Commands and attributes required to tunnel the BACnet protocol.
0x0602	Analog input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog measurement.
0x0603	Analog input (BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog measurement.
0x0604	Analog output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog output.
0x0605	Analog output (BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog output.
0x0606	Analog value(BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of an analog value, typically used as a control system parameter.
0x0607	Analog value(BACnet extended)	An interface for accessing a number of BACnet based attributes of an analog value, typically used as a control system parameter.

Cluster ID	Cluster Name	Description
0x0608	Binary input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary measurement.
0x0609	Binary input (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary measurement.
0x060a	Binary output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary output.
0x060b	Binary output (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary output.
0x060c	Binary value (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a binary value, typically used as a control system parameter.
0x060d	Binary value (BACnet extended)	An interface for accessing a number of BACnet based attributes of a binary value, typically used as a control system parameter.
0x060e	Multistate input (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate measurement.
0x060f	Multistate input (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate measurement.
0x0610	Multistate output (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate output.
0x0611	Multistate output (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate output.
0x0612	Multistate value (BACnet regular)	An interface for accessing a number of commonly used BACnet based attributes of a multistate value, typically used as a control system parameter.
0x0613	Multistate value (BACnet extended)	An interface for accessing a number of BACnet based attributes of a multistate value, typically used as a control system parameter.
0x0614	11073 Protocol Tunnel	Interface for 11073 Protocol Tunnel used in health care applications
0x0615	ISO7816 Tunnel	Commands and attributes for mobile office solutions using devices.

11495 **9.2 Generic Tunnel**

11496 **9.2.1 Overview**

11497 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 11498 etc.

11499 The generic cluster provides the minimum common commands and attributes required to discover protocol tunnelling  
 11500 devices. A protocol cluster specific to the protocol being tunneled shall be implemented on the same endpoint as the  
 11501 Generic Tunnel cluster.

11502 **Note:** The reverse is not true, as there may be tunnel clusters that do not require the Generic Tunnel cluster.

11503 **9.2.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11504 **9.2.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TUN	Type 1 (client to server)

11505 **9.2.1.3 Cluster Identifiers**

Identifier	Name
0x0600	Generic Tunnel

11506 **9.2.2 Server**

11507 **9.2.2.1 Dependencies**

11508 The maximum size of the *ProtocolAddress* attribute is dependent on the protocol supported by any associated specific  
 11509 protocol tunnel cluster supported on the same endpoint (see 9.2.2.2.3, ProtocolAddress Attribute).

11510 **9.2.2.2 Attributes**

11511 The Generic Tunnel contains the attributes summarized in Table 9-2.

11512 **Table 9-2. Attributes of the Generic Tunnel Cluster**

Id	Name	Type	Range	Acc	Default	M/O
0x0001	MaximumIncomingTransferSize	uint16	0x0000 - 0xffff	R	0x0000	M
0x0002	MaximumOutgoingTransferSize	uint16	0x0000 - 0xffff	R	0x0000	M
0x0003	ProtocolAddress	octstr	0 - 255 octets	RW	Null string	M

**11513 9.2.2.2.1 MaximumIncomingTransferSize Attribute**

11514 The *MaximumIncomingTransferSize* attribute specifies the maximum size, in octets, of the application service data  
11515 unit (ASDU) that can be transferred to this node in one single message transfer. The ASDU referred to is the ZCL  
11516 frame, including header and payload, of any command received by a protocol specific tunnel cluster on the same  
11517 endpoint.

11518 This value cannot exceed the Maximum Incoming Transfer Size field of the node descriptor on the device supporting  
11519 this cluster.

**11520 9.2.2.2.2 MaximumOutgoingTransferSize Attribute**

11521 The *MaximumOutgoingTransferSize* attribute specifies the maximum size, in octets, of the application sub-layer data  
11522 unit (ASDU) that can be transferred from this node in one single message transfer. The ASDU referred to is the ZCL  
11523 frame, including header and payload, of any command sent by a protocol specific tunnel cluster on the same endpoint.

11524 This value cannot exceed the Maximum Outgoing Transfer Size field of the node descriptor on the device supporting  
11525 this cluster.

**11526 9.2.2.2.3 ProtocolAddress Attribute**

11527 The *ProtocolAddress* attribute contains an octet string that is interpreted as a device address by the protocol being  
11528 tunneled by an associated protocol specific tunnel cluster (if any). The overall maximum size of the string is 255  
11529 octets, but devices need only support the actual maximum size required by that protocol

**11530 9.2.2.3 Commands Received**

11531 The cluster specific commands received by the Generic Tunnel server cluster are listed in Table 9-3.

11532 **Table 9-3. Command IDs Received by the Generic Tunnel Cluster**

Identifier	Description	M/O
0x00	Match Protocol Address	M

**11533 9.2.2.3.1 Match Protocol Address Command**

11534 The Match Protocol Address command payload shall be formatted as illustrated in Figure 9-1.

11535 **Figure 9-1. Format of Match Protocol Address Command Payload**

octets	Variable
Data Type	octstr
Field Name	Protocol Address

**11536 9.2.2.3.2 When Generated**

11537 This command is generated when an associated protocol specific tunnel cluster wishes to find the address (node,  
11538 endpoint) of the Generic Tunnel server cluster representing a protocol-specific device with a given protocol address.  
11539 The command is typically multicast to a group of inter-communicating Generic Tunnel clusters.

11540 **9.2.2.3.3 Effect on Receipt**

11541 On receipt of this command, a device shall match the Protocol Address field of the received command to the  
 11542 ProtocolAddress attribute. If they are equal, it shall return the Match Protocol Address Response command (see  
 11543 9.2.2.4.1), otherwise it shall do nothing.

11544 **9.2.2.4 Commands Generated**

11545 The cluster specific commands generated by the Generic Tunnel server cluster are listed in Table 9-4. Command IDs  
 11546 Generated by the Generic Tunnel Cluster.

11547 **Table 9-4. Command IDs Generated by the Generic Tunnel Cluster**

Identifier	Description	M/O
0x00	Match Protocol Address Response	M
0x01	Advertise Protocol Address	O

11548

11549 **9.2.2.4.1 Match Protocol Address Response Command**

11550 The Match Protocol Address Response command payload shall be formatted as illustrated in Figure 9-2.

11551 **Figure 9-2. Match Protocol Address Response Command Payload**

octets	8	Variable
Data Type	EUI64	octstr
Field Name	Device IEEE Address	Protocol Address

11552

11553 The Device IEEE Address field shall be set equal to the IEEE address of the responding device. The Protocol Address  
 11554 field shall be set equal to the matched Protocol Address.

11555 **9.2.2.4.2 When Generated**

11556 This command is generated upon receipt of a Match Protocol Address command (see 9.2.2.3.1), to indicate that the  
 11557 Protocol Address was successfully matched by the responding device.

11558 **9.2.2.4.3 Advertise Protocol Address Command**

11559 The Advertise Protocol Address command payload shall be formatted as illustrated in Figure 9-3.

11560 **Figure 9-3. Advertise Protocol Address Command Payload**

octets	Variable
Data Type	octstr
Field Name	Protocol Address

11561

11562 The Protocol Address field shall be set to the value of the *ProtocolAddress* attribute.

11563 **9.2.2.4.4 When Generated**

11564 This command is typically sent upon startup, and whenever the *ProtocolAddress* attribute changes. It is typically  
11565 multicast to a group of inter-communicating Generic Tunnel clusters.

11566 **9.2.3 Client**

11567 The client cluster has no specific attributes or dependencies. The client cluster receives the cluster specific commands  
11568 detailed in Commands Generated. The client cluster generates the cluster specific commands detailed in 9.2.2.3.

11569 **9.3 BACnet Protocol Tunnel**11570 **9.3.1 Overview**

11571 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
11572 etc.

11573 The BACnet Protocol Tunnel cluster provides the commands and attributes required to tunnel the BACnet protocol  
11574 (see [A1]). The server cluster receives BACnet NPDUs and the client cluster generates BACnet NPDUs, thus it is  
11575 necessary to have both server and client on an endpoint to tunnel BACnet messages in both directions.

11576 **9.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11577 **9.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BACTUN	Type 1 (client to server)

11578 **9.3.1.3 Cluster Identifiers**

Identifier	Name
0x0601	BACnet Protocol Tunnel

11579 **9.3.2 Server**11580 **9.3.2.1 Dependencies**

11581 Any endpoint that supports the BACnet Protocol Tunnel server cluster shall also support the Generic Tunnel server  
11582 cluster.

11583 The associated Generic Tunnel server cluster shall have its *ProtocolAddress* attribute equal to the device identifier of  
11584 the BACnet device represented on that endpoint, expressed as an octet string (i.e., with identical format as a BACnet  
11585 OID data type, but interpreted as an octet string). The special three octet value 0x3FFFFFF of the *ProtocolAddress*  
11586 attribute indicates that the associated BACnet device is not commissioned.



11587 The associated Generic Tunnel server cluster shall also have its *MaximumIncomingTransferSize* attribute and  
 11588 *MaximumOutgoingTransferSize* attribute equal to or greater than 504 octets. Accordingly, this cluster requires  
 11589 fragmentation to be implemented, with maximum transfer sizes given by these attributes.

11590 **9.3.2.2 Attributes**

11591 The BACnet Protocol Tunnel cluster does not contain any attributes.

11592 **9.3.2.3 Commands Received**

11593 The cluster specific commands received by the BACnet Protocol Tunnel server cluster are listed in Table 9-5.

11594 **Table 9-5. Command IDs for the BACnet Protocol Tunnel Cluster**

Identifier	Description	M/O
0x00	Transfer NPDU	M

11595 **9.3.2.3.1 Transfer NPDU Command**

11596 **9.3.2.3.1.1 Payload Format**

11597 The Transfer NPDU command payload shall be formatted as illustrated in Figure 9-4.

11598 **Figure 9-4. Format of the Transfer NPDU Command Payload**

octets	Variable
Data Type	Sequence of data8
Field Name	NPDU

11599 **9.3.2.3.1.2 NPDU Field**

11600 The NPDU field is of variable length and is a BACnet NPDU as defined in the BACnet standard [A1]. Its format is a  
 11601 sequence of 8-bit data (see General Data section of Chapter 2 of arbitrary length).

11602 **9.3.2.3.1.3 When Generated**

11603 This command is generated when a BACnet network layer wishes to transfer a BACnet NPDU across a tunnel to  
 11604 another BACnet network layer.

11605 **9.3.2.3.1.4 Effect on Receipt**

11606 On receipt of this command, a device shall process the BACnet NPDU as specified in the BACnet standard [A1].

11607 **9.3.2.4 Commands Generated**

11608 No cluster specific commands are generated by the server cluster.

11609 **9.3.3 Client**

11610 The client cluster has no specific attributes or dependencies. The client does not receive any cluster specific  
 11611 commands. The cluster specific commands generated by the client cluster are listed in 9.3.2.3.

## 11612 9.4 BACnet Input, Output and Value Clusters

### 11613 9.4.1 Overview

11614 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
11615 etc.

11616 This section specifies a number of clusters which are based on the Input, Output and Value objects specified by  
11617 BACnet (see [A1]).

11618 Each of these three objects is specified by BACnet in three different forms - Analog, Binary and Multistate. clusters  
11619 are specified here based on all nine such BACnet objects.

11620 Each such BACnet object is represented in the ZCL by three related clusters – a BACnet Basic cluster , a BACnet  
11621 Regular cluster and a BACnet Extended cluster. The properties of each BACnet object are implemented as attributes,  
11622 and are divided into three sets, which are allocated to the clusters as follows.

11623 BACnet Basic clusters implement attributes and functionality that can be readily employed either via interworking  
11624 with a BACnet system, or by a non-BACnet system. Accordingly, these clusters are included in the ZCL General  
11625 functional domain.

11626 BACnet Regular and BACnet Extended clusters implement attributes and functionality that are specifically intended  
11627 for interworking with a BACnet system (through a BACnet gateway). Accordingly, these clusters are included in the  
11628 ZCL Protocol Interface functional domain.

11629 A BACnet Regular cluster may only be implemented on an endpoint that also implements its associated Basic cluster.  
11630 Similarly, a BACnet Extended cluster may only be implemented on an endpoint that also implements both its  
11631 associated BACnet Regular cluster and its associated Basic cluster.

11632 The clusters specified herein are for use typically in Commercial Building applications, but may be used in any  
11633 application domain.

### 11634 9.4.2 Analog Input (BACnet Regular)

11635 The Analog Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet  
11636 based attributes of an analog measurement. It is used principally for interworking with BACnet systems.

#### 11637 9.4.2.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

#### 11638 9.4.2.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BAI	Type 2 (server to client)

#### 11639 9.4.2.3 Cluster Identifiers

Identifier	Name
0x0602	Analog Input (BACnet Regular)

11640 **9.4.2.4 Server**

11641 **9.4.2.4.1 Dependencies**

11642 Any endpoint that supports this cluster must support the Analog Input (Basic) cluster.

11643 **9.4.2.4.2 Attributes**

11644 The attributes of this cluster are detailed in Table 9-6.

11645 **Table 9-6. Attributes of the Analog Input (BACnet Regular) Server**

Identifier	Name	Type	Range	Acc	Default	M/O
0x0016	<i>COVIncrement</i>	single	-	R*W	0	O
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000-0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x0076	<i>UpdateInterval</i>	uint8	-	R*W	0	O
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11646

11647 For an explanation of the attributes, see section 9.4.20.

11648 **9.4.2.4.3 Commands**

11649 No cluster specific commands are received or generated.

11650 **9.4.2.4.4 Attribute Reporting**

11651 No attribute reporting is mandated for this cluster.

11652 **9.4.2.5 Client**

11653 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11654 **9.4.3 Analog Input (BACnet Extended)**

11655 The Analog Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes  
 11656 of an analog measurement. It is used principally for interworking with BACnet systems.

11657 **9.4.3.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11658 **9.4.3.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AIBE	Type 2 (server to client)

11659 **9.4.3.3 Cluster Identifiers**

Identifier	Name
0x0603	Analog Input (BACnet Extended)

11660 **9.4.3.4 Server**11661 **9.4.3.4.1 Dependencies**

11662 Any endpoint that supports this cluster must support the Analog Input (Basic) cluster and the Analog Input (BACnet  
11663 Regular) cluster.

11664 **9.4.3.4.2 Attributes**

11665 The attributes of this cluster are detailed in Table 9-7.

11666 **Table 9-7. Attributes of the Analog Input (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	<i>Deadband</i>	single	-	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x002D	<i>HighLimit</i>	single	-	R*W	0	M
0x0034	<i>LimitEnable</i>	map8	0x00 - 0x11	R*W	0x00	M
0x003B	<i>LowLimit</i>	single	-	R*W	0	M
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11667

11668 For an explanation of the attributes, see section 9.4.20 and 9.4.21.

11669 **9.4.3.4.3 Commands**

11670 No cluster specific commands are received or generated.

11671 **9.4.3.5 Client**

11672 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11673 **9.4.4 Analog Output (BACnet Regular)**

11674 The Analog Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet  
 11675 based attributes of an analog output. It is used principally for interworking with BACnet systems.

11676 **9.4.4.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11677 **9.4.4.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AOB	Type 2 (server to client)

11678 **9.4.4.3 Cluster Identifiers**

Identifier	Name
0x0604	Analog Output (BACnet Regular)

11679 **9.4.4.4 Server**

11680 **9.4.4.4.1 Dependencies**

11681 Any endpoint that supports this cluster shall also support the Analog Output (Basic) cluster, and this cluster shall  
 11682 support the *PriorityArray* and *RelinquishDefault* attributes.

11683 **9.4.4.4.2 Attributes**

11684 The attributes of this cluster are detailed in Table 9-8.

11685 **Table 9-8. Attributes of the Analog Output (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	M/O
0x0016	<i>COVIncrement</i>	single	-	R*W	0	O
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11686

11687 For an explanation of the attributes, see section 9.4.20.

11688 **9.4.4.4.3 Commands**

11689 No cluster specific commands are received or generated.

11690 **9.4.4.4.4 Attribute Reporting**

11691 No attribute reporting is mandated for this cluster.

11692 **9.4.4.5 Client**

11693 The client has no dependencies, no specific attributes, and receives or generates no cluster specific commands.

11694 **9.4.5 Analog Output (BACnet Extended)**11695 The Analog Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based  
11696 attributes of an analog output. It is used principally for interworking with BACnet systems.11697 **9.4.5.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11698 **9.4.5.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AOBE	Type 2 (server to client)

11699 **9.4.5.3 Cluster Identifiers**

Identifier	Name
0x0605	Analog Output (BACnet Extended)

11700 **9.4.5.4 Server**11701 **9.4.5.4.1 Dependencies**11702 Any endpoint that supports this cluster must support the Analog Output (Basic) cluster and the Analog Output  
11703 (BACnet Regular) cluster.11704 **9.4.5.4.2 Attributes**

11705 The attributes of this cluster are detailed in Table 9-9.

11706

**Table 9-9. Attributes of the Analog Output (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	<i>Deadband</i>	single	-	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x002D	<i>HighLimit</i>	single	-	R*W	0	M
0x0034	<i>LimitEnable</i>	map8	0x00 - 0x11	R*W	0x00	M
0x003B	<i>LowLimit</i>	single	-	R*W	0	M
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11707

11708 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11709 **9.4.5.4.3 Commands**

11710 No cluster specific commands are received or generated.

11711 **9.4.5.4.4 Attribute Reporting**

11712 No attribute reporting is mandated for this cluster.

11713 **9.4.5.5 Client**

11714 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11715 **9.4.6 Analog Value (BACnet Regular)**

11716 The Analog Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based  
 11717 characteristics of an analog value, typically used as a control system parameter. It is principally used for interworking  
 11718 with BACnet systems.

11719 **9.4.6.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11720 **9.4.6.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AVB	Type 2 (server to client)

11721 **9.4.6.3 Cluster Identifiers**

Identifier	Name
0x0606	Analog Value (BACnet Regular)

11722 **9.4.6.4 Server**

11723 **9.4.6.4.1 Dependencies**

11724 Any endpoint that supports this cluster must support the Analog Value (Basic) cluster.

11725 **9.4.6.4.2 Attributes**

11726 The attributes of this cluster are detailed in Table 9-10.

11727 **Table 9-10. Attributes of the Analog Value (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	M/O
0x0016	<i>COVIncrement</i>	single	-	R*W	0	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11728  
11729 For an explanation of the attributes, see section 9.4.20.

11730 **9.4.6.4.3 Commands**

11731 No cluster specific commands are received or generated.

11732 **9.4.6.5 Client**

11733 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11734 **9.4.7 Analog Value (BACnet Extended)**

11735 The Analog Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of an  
11736 analog value, typically used as a control system parameter. It is principally used for interworking with BACnet  
11737 systems.



11738 **9.4.7.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11739 **9.4.7.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	AVBE	Type 2 (server to client)

11740 **9.4.7.3 Cluster Identifiers**

Identifier	Name
0x0607	Analog Value (BACnet Extended)

11741 **9.4.7.4 Server**

11742 **9.4.7.4.1 Dependencies**

11743 Any endpoint that supports this cluster must support the Analog Value (Basic) cluster and the Analog Value (BACnet  
 11744 Regular) cluster.

11745 **9.4.7.4.2 Attributes**

11746 The attributes of this cluster are detailed in Table 9-11.

11747 **Table 9-11. Attributes of the Analog Value (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0019	<i>Deadband</i>	single	-	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x002D	<i>HighLimit</i>	single	-	R*W	0	M
0x0034	<i>LimitEnable</i>	map8	0x00 - 0x11	R*W	0x00	M
0x003B	<i>LowLimit</i>	single	-	R*W	0	M
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11748

11749 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

#### 11750 **9.4.7.4.3 Commands**

11751 No cluster specific commands are received or generated.

#### 11752 **9.4.7.5 Client**

11753 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

### 11754 **9.4.8 Binary Input (BACnet Regular)**

11755 The Binary Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet  
11756 based attributes of a binary measurement. It is used principally for interworking with BACnet systems.

#### 11757 **9.4.8.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

#### 11758 **9.4.8.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BIB	Type 2 (server to client)

#### 11759 **9.4.8.3 Cluster Identifiers**

Identifier	Name
0x0608	Binary Input (BACnet Regular)

#### 11760 **9.4.8.4 Server**

##### 11761 **9.4.8.4.1 Dependencies**

11762 Any endpoint that supports this cluster must support the Binary Input (Basic) cluster.

##### 11763 **9.4.8.4.2 Attributes**

11764 The attributes of this cluster are detailed in Table 9-12.

11765 **Table 9-12. Attributes of the Binary Input (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	MO
0x000F	<i>ChangeOfStateCount</i>	uint32	-	R*W	0xffffffff	O
0x0010	<i>ChangeOfStateTime</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Default</b>	<b>MO</b>
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x0021	<i>ElapsedActiveTime</i>	uint32	-	R*W	0xffffffff	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x0072	<i>TimeOfATReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	<i>TimeOfSCReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11766

11767 For an explanation of the attributes, see section 9.4.20.

11768 **9.4.8.4.3 Commands**

11769 No cluster specific commands are received or generated.

11770 **9.4.8.5 Client**

11771 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11772 **9.4.9 Binary Input (BACnet Extended)**

11773 The Binary Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based attributes  
 11774 of a binary measurement. It is used principally for interworking with BACnet systems.

11775 **9.4.9.1 Revision History**

<b>Rev</b>	<b>Description</b>
1	mandatory global <i>ClusterRevision</i> attribute added

11776 **9.4.9.2 Classification**

<b>Hierarchy</b>	<b>Role</b>	<b>PICS Code</b>	<b>Primary Transaction</b>
Base	Application	BIBE	Type 2 (server to client)

11777 **9.4.9.3 Cluster Identifiers**

Identifier	Name
0x0609	Binary Input (BACnet Extended)

11778 **9.4.9.4 Server**11779 **9.4.9.4.1 Dependencies**

11780 Any endpoint that supports this cluster must support the Binary Input (Basic) cluster and the Binary Input (BACnet  
11781 Regular) cluster.

11782 **9.4.9.4.2 Attributes**

11783 The attributes of this cluster are detailed in Table 9-13.

11784 **Table 9-13. Attributes of the Binary Input (BACnet Extended) Server**

Id	Name	Type	Range	Access	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0006	<i>AlarmValue</i>	bool	0 - 1	R*W	-	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11785  
11786 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11787 **9.4.9.4.3 Commands**

11788 No cluster specific commands are received or generated.

11789 **9.4.9.5 Client**

11790 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11791 **9.4.10 Binary Output (BACnet Regular)**

11792 The Analog Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used BACnet  
 11793 based attributes of a binary output. It is used principally for interworking with BACnet systems.

11794 **9.4.10.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11795 **9.4.10.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BOB	Type 2 (server to client)

11796 **9.4.10.3 Cluster Identifiers**

Identifier	Name
0x060a	Binary Output (BACnet Regular)

11797 **9.4.10.4 Server**

11798 **9.4.10.4.1 Dependencies**

11799 Any endpoint that supports this cluster shall also support the Binary Output (Basic) cluster, and this cluster shall  
 11800 support the PriorityArray and RelinquishDefault attributes.

11801 **9.4.10.4.2 Attributes**

11802 The attributes of this cluster are detailed in Table 9-14.

11803 **Table 9-14. Attributes of the Binary Output (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	MO
0x000F	<i>ChangeOfStateCount</i>	uint32	-	R*W	0xffffffff	O
0x0010	<i>ChangeOfStateTime</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x0021	<i>ElapsedActiveTime</i>	uint32	-	R*W	0xffffffff	O
0x0028	<i>FeedBackValue</i>	enum8	0 - 1	R*W	0	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000 - 0xffffffff	R	-	M

Id	Name	Type	Range	Acc	Default	MO
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x0072	<i>TimeOfATReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	<i>TimeOfSCReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11804

11805 For an explanation of the attributes, see section 9.4.20.

11806 **9.4.10.4.3 Commands**

11807 No cluster specific commands are received or generated.

11808 **9.4.10.4.4 Attribute Reporting**

11809 No attribute reporting is mandated for this cluster.

11810 **9.4.10.5 Client**

11811 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11812 **9.4.11 Binary Output (BACnet Extended)**11813 The Binary Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based  
11814 attributes of a binary output. It is used principally for interworking with BACnet systems.11815 **9.4.11.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11816 **9.4.11.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BOBE	Type 2 (server to client)

11817 **9.4.11.3 Cluster Identifiers**

Identifier	Name
0x060b	Binary Output (BACnet Extended)

11818 **9.4.11.4 Server**

11819 **9.4.11.4.1 Dependencies**

11820 Any endpoint that supports this cluster must support the Binary Output (Basic) cluster and the Binary Output (BACnet  
 11821 Regular) cluster.

11822 **9.4.11.4.2 Attributes**

11823 The attributes of this cluster are detailed in Table 9-15.

11824 **Table 9-15. Attributes of the Binary Output (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11825

11826 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11827 **9.4.11.4.3 Commands**

11828 No cluster specific commands are received or generated.

11829 **9.4.11.5 Client**

11830 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11831 **9.4.12 Binary Value (BACnet Regular)**

11832 The Binary Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based  
 11833 characteristics of a binary value, typically used as a control system parameter. It is principally used for interworking  
 11834 with BACnet systems.

11835 **9.4.12.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11836 **9.4.12.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BVB	Type 2 (server to client)

11837 **9.4.12.3 Cluster Identifiers**

Identifier	Name
0x060c	Binary Value (BACnet Regular)

11838 **9.4.12.4 Server**11839 **9.4.12.4.1 Dependencies**

11840 Any endpoint that supports this cluster must support the Binary Value (Basic) cluster.

11841 **9.4.12.4.2 Attributes**

11842 The attributes of this cluster are detailed in Table 9-16.

11843 **Table 9-16. Attributes of the Binary Value (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	M/O
0x000F	<i>ChangeOfStateCount</i>	uint32	-	R*W	0xffffffff	O
0x0010	<i>ChangeOfStateTime</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0021	<i>ElapsedActiveTime</i>	uint32	-	R*W	0xffffffff	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0-0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x0072	<i>TimeOfATReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x0073	<i>TimeOfSCReset</i>	struct (date, ToD)	-	R	0xffffffff 0xffffffff	O
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11844

11845 For an explanation of the attributes, see section 9.4.20.

11846 **9.4.12.4.3 Commands**

11847 No cluster specific commands are received or generated.



11848 **9.4.12.4.4 Attribute Reporting**

11849 No attribute reporting is mandated for this cluster.

11850 **9.4.12.5 Client**

11851 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11852 **9.4.13 Binary Value (BACnet Extended)**

11853 The Binary Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of a  
 11854 binary value, typically used as a control system parameter. It is principally used for interworking with BACnet  
 11855 systems.

11856 **9.4.13.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11857 **9.4.13.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	BVBE	Type 2 (server to client)

11858 **9.4.13.3 Cluster Identifiers**

Identifier	Name
0x060d	Binary Value (BACnet Extended)

11859 **9.4.13.4 Server**

11860 **9.4.13.4.1 Dependencies**

11861 Any endpoint that supports this cluster must support the Binary Value (Basic) cluster and the Binary Value (BACnet  
 11862 Regular) cluster.

11863 **9.4.13.4.2 Attributes**

11864 The attributes of this cluster are detailed in Table 9-17.

11865 **Table 9-17. Attributes of the Binary Value (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0006	<i>AlarmValue</i>	bool	0 - 1	R*W	-	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M

Id	Name	Type	Range	Acc	Def	M/O
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11866

11867 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11868 **9.4.13.4.3 Commands**

11869 No cluster specific commands are received or generated.

11870 **9.4.13.5 Client**

11871 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11872 **9.4.14 Multistate Input (BACnet Regular)**11873 The Multistate Input (BACnet Regular) cluster provides an interface for accessing a number of commonly used  
11874 BACnet based attributes of a multistate measurement. It is used principally for interworking with BACnet systems.11875 **9.4.14.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11876 **9.4.14.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MIB	Type 2 (server to client)

11877 **9.4.14.3 Cluster Identifiers**

Identifier	Name
0x060e	Multistate Input (BACnet Regular)

11878 **9.4.14.4 Server**11879 **9.4.14.4.1 Dependencies**

11880 Any endpoint that supports this cluster must support the Multistate Input (Basic) cluster.

11881 **9.4.14.4.2 Attributes**

11882 The attributes of this cluster are detailed in Table 9-18.

11883 **Table 9-18. Attributes of the Multistate Input (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	M/O
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0-0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11884

11885 For an explanation of the attributes, see section 9.4.20.

11886 **9.4.14.4.3 Commands**

11887 No cluster specific commands are received or generated.

11888 **9.4.14.5 Client**

11889 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11890 **9.4.15 Multistate Input (BACnet Extended)**

11891 The Multistate Input (BACnet Extended) cluster provides an interface for accessing a number of BACnet based  
 11892 attributes of a multistate measurement. It is used principally for interworking with BACnet systems.

11893 **9.4.15.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11894 **9.4.15.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MIBE	Type 2 (server to client)

11895 **9.4.15.3 Cluster Identifiers**

Identifier	Name
------------	------

0x060f	Multistate Input (BACnet Extended)
--------	------------------------------------

11896 **9.4.15.4 Server**11897 **9.4.15.4.1 Dependencies**

11898 Any endpoint that supports this cluster must support the Multistate Input (Basic) cluster and the Multistate Input  
11899 (BACnet Regular) cluster.

11900 **9.4.15.4.2 Attributes**

11901 The attributes of this cluster are detailed in Table 9-19.

11902 **Table 9-19. Attributes of Multistate Input (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0006	<i>AlarmValues</i>	Set of uint16	0 - 0xffff	R*W	-	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0025	<i>FaultValues</i>	Set of uint16	0 - 0xffff	R*W	0	M
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11903

11904 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11905 **9.4.15.4.3 Commands**

11906 No cluster specific commands are received or generated.

11907 **9.4.15.4.4 Attribute Reporting**

11908 No attribute reporting is mandated for this cluster.

11909 **9.4.15.5 Client**

11910 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11911 **9.4.16 Multistate Output (BACnet Regular)**

11912 The Multistate Output (BACnet Regular) cluster provides an interface for accessing a number of commonly used  
11913 BACnet based attributes of a multistate output. It is used principally for interworking with BACnet systems.

11914 **9.4.16.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11915 **9.4.16.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MOB	Type 2 (server to client)

11916 **9.4.16.3 Cluster Identifiers**

Identifier	Name
0x0610	Multistate Output (BACnet Regular)

11917 **9.4.16.4 Server**

11918 **9.4.16.4.1 Dependencies**

11919 Any endpoint that supports this cluster shall also support the Multistate Output (Basic) cluster, and this cluster shall  
 11920 support the PriorityArray and RelinquishDefault attributes.

11921 **9.4.16.4.2 Attributes**

11922 The attributes of this cluster are detailed in Table 9-20.

11923 **Table 9-20. Attributes of Multistate Output (BACnet Regular) Server**

Id	Name	Type	Range	Access	Default	M/O
0x001F	<i>DeviceType</i>	string	-	R	Null string	O
0x0028	<i>FeedBackValue</i>	enum8	0 - 1	R*W	0	O
0x004B	<i>ObjectIdentifier</i>	bacOID	0x00000000 - 0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11924  
 11925 For an explanation of the attributes, see section 9.4.20.

11926 **9.4.16.4.3 Commands**

11927 No cluster specific commands are received or generated.

11928 **9.4.16.5 Client**

11929 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11930 **9.4.17 Multistate Output (BACnet Extended)**11931 The Multistate Output (BACnet Extended) cluster provides an interface for accessing a number of BACnet based  
11932 attributes of a multistate output. It is used principally for interworking with BACnet systems.11933 **9.4.17.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11934 **9.4.17.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MOBE	Type 2 (server to client)

11935 **9.4.17.3 Cluster Identifiers**

Identifier	Name
0x0611	Multistate Output (BACnet Extended)

11936 **9.4.17.4 Server**11937 **9.4.17.4.1 Dependencies**11938 Any endpoint that supports this cluster must support the Multistate Output (Basic) cluster and the Multistate Output  
11939 (BACnet Regular) cluster.11940 **9.4.17.4.2 Attributes**

11941 The attributes of this cluster are detailed in Table 9-21.

11942 **Table 9-21. Attributes of Multistate Output (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - 0xffff	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M

Id	Name	Type	Range	Acc	Def	M/O
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11943

11944 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11945 **9.4.17.4.3 Commands**

11946 No cluster specific commands are received or generated.

11947 **9.4.17.4.4 Attribute Reporting**

11948 No attribute reporting is mandated for this cluster.

11949 **9.4.17.5 Client**

11950 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11951 **9.4.18 Multistate Value (BACnet Regular)**

11952 The Multistate Value (BACnet Regular) cluster provides an interface for accessing commonly used BACnet based characteristics of a multistate value, typically used as a control system parameter. It is principally used for interworking with BACnet systems.

11955 **9.4.18.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11956 **9.4.18.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MVB	Type 2 (server to client)

11957 **9.4.18.3 Cluster Identifiers**

Identifier	Name
0x0612	Multistate Value (BACnet Regular)

11958 **9.4.18.4 Server**

11959 **9.4.18.4.1 Dependencies**

11960 Any endpoint that supports this cluster must support the Multistate Value (Basic) cluster.

11961 **9.4.18.4.2 Attributes**

11962 The attributes of this cluster are detailed in Table 9-22.

11963 **Table 9-22. Attributes of Multistate Value (BACnet Regular) Server**

Id	Name	Type	Range	Acc	Default	M/O
0x004B	<i>ObjectIdentifier</i>	bacOID	0 -0xffffffff	R	-	M
0x004D	<i>ObjectName</i>	string	-	R	Null string	M
0x004F	<i>ObjectType</i>	enum16	-	R	-	M
0x00A8	<i>ProfileName</i>	string	-	R*W	Null string	O

11964

11965 For an explanation of the attributes, see section 9.4.20.

11966 **9.4.18.4.3 Commands**

11967 No cluster specific commands are received or generated.

11968 **9.4.18.5 Client**

11969 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11970 **9.4.19 Multistate Value (BACnet Extended)**

11971 The Multistate Value (BACnet Extended) cluster provides an interface for accessing BACnet based characteristics of  
 11972 a multistate value, typically used as a control system parameter. It is principally used for interworking with BACnet  
 11973 systems.

11974 **9.4.19.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

11975 **9.4.19.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	MVBE	Type 2 (server to client)

11976 **9.4.19.3 Cluster Identifiers**

Identifier	Name
0x0613	Multistate Value (BACnet Extended)



11977 **9.4.19.4 Server**

11978 **9.4.19.4.1 Dependencies**

11979 Any endpoint that supports this cluster must support the Multistate Value (Basic) cluster and the Multistate Value  
 11980 (BACnet Regular) cluster.

11981 **9.4.19.4.2 Attributes**

11982 The attributes of this cluster are detailed in Table 9-23.

11983 **Table 9-23. Attributes of Multistate Value (BACnet Extended) Server**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>AckedTransitions</i>	map8	-	R*W	0	M
0x0006	<i>AlarmValues</i>	set of uint16	0 - 0xffff	R*W	-	M
0x0011	<i>NotificationClass</i>	uint16	0x0000 - xffff	R*W	0	M
0x0023	<i>EventEnable</i>	map8	-	R*W	0	M
0x0024	<i>EventState</i>	enum8	-	R	0	O
0x0025	<i>FaultValues</i>	set of uint16	0 - 0xffff	R*W	0	M
0x0048	<i>NotifyType</i>	enum8	-	R*W	0	M
0x0071	<i>TimeDelay</i>	uint8	-	R*W	0	M
0x0082	<i>EventTimeStamps</i>	array[3] of (uint16, ToD, or struct of (date, ToD))	-	R	-	M

11984

11985 For an explanation of the attributes, see sections 9.4.20 and 9.4.21.

11986 **9.4.19.4.3 Commands**

11987 No cluster specific commands are received or generated.

11988 **9.4.19.4.4 Attribute Reporting**

11989 No attribute reporting is mandated for this cluster.

11990 **9.4.19.5 Client**

11991 The client has no dependencies, no attributes, and receives or generates no cluster specific commands.

11992 **9.4.20 Attributes of BACnet Regular Clusters**

11993 The attributes of BACnet Regular and BACnet Extended clusters are specifically intended for interworking with  
 11994 BACnet systems (via a BACnet gateway). They are based on BACnet properties with the same names. See the BACnet  
 11995 Reference Manual [A1] for detailed descriptions of these properties.

11996 References to reports in this section refer to BACnet intrinsic reporting. Note that attribute reporting may be used to  
 11997 send reports as well.

### 11998 **9.4.20.1 ObjectIdentifier Attribute**

11999 This attribute, of type BACnet OID, is a numeric code that is used to identify the object. It shall be unique within the  
12000 BACnet Device that maintains it.

### 12001 **9.4.20.2 ObjectName Attribute**

12002 This attribute, of type Character String, shall represent a name for the object that is unique within the BACnet Device  
12003 that maintains it. The minimum length of the string shall be one character. The set of characters used in the *ObjectName*  
12004 shall be restricted to printable characters.

### 12005 **9.4.20.3 ObjectType Attribute**

12006 This attribute, of type enumeration, is set to the ID of the corresponding BACnet object type from which the cluster  
12007 was derived.

### 12008 **9.4.20.4 COVIncrement Attribute**

12009 This attribute, of type single, specifies the minimum change in *PresentValue* that will cause a value change report to  
12010 be initiated to bound report recipient clients. This value is the same as the Reportable Change value for the  
12011 *PresentValue* attribute.

### 12012 **9.4.20.5 DeviceType Attribute**

12013 This attribute, of type Character String, is a text description of the physical device connected to the input, output or  
12014 value.

### 12015 **9.4.20.6 UpdateInterval Attribute**

12016 This attribute indicates the maximum period of time between updates to the *PresentValue* of an Analog Input cluster,  
12017 in hundredths of a second, when the input is not overridden and not out-of-service.

### 12018 **9.4.20.7 ChangeOfStateCount Attribute**

12019 This attribute, of type Unsigned 32-bit integer, represents the number of times that the *PresentValue* attribute of a  
12020 Binary Input, Output or Value cluster has changed state (from 0 to 1, or from 1 to 0) since the *ChangeOfStateCount*  
12021 attribute was most recently set to a zero value. The *ChangeOfStateCount* attribute shall have a range of 0-65535 or  
12022 greater.

12023 When *OutOfService* is FALSE, a change to the *Polarity* attribute shall alter *PresentValue* and thus be considered a  
12024 change of state. When *OutOfService* is TRUE, changes to *Polarity* shall not cause changes of state. If one of the  
12025 optional attributes *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfStateCountReset* is present, then all of these  
12026 attributes shall be present.

### 12027 **9.4.20.8 ChangeOfStateTime Attribute**

12028 This attribute, of type Structure (Date, Time of Day), represents the most recent date and time at which the  
12029 *PresentValue* attribute of a Binary Input, Output or Value cluster changed state (from 0 to 1, or from 1 to 0)

12030 When *OutOfService* is FALSE, a change to the *Polarity* attribute shall alter *PresentValue* and thus be considered a  
12031 change of state. When *OutOfService* is TRUE, changes to *Polarity* shall not cause changes of state. If one of the  
12032 optional attributes *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfSCReset* is present, then all of these attributes  
12033 shall be present.

### 12034 **9.4.20.9 ElapsedActiveTime Attribute**

12035 This attribute, of type Unsigned 32-bit integer, represents the accumulated number of seconds that the *PresentValue*  
12036 attribute of a Binary Input, Output or Value cluster has had the value ACTIVE (1) since the *ElapsedActiveTime*  
12037 attribute was most recently set to a zero value. If one of the optional properties *ElapsedActiveTime* or *TimeOfATReset*  
12038 is present, then both of these attributes shall be present.

### 12039 **9.4.20.10 TimeOfATReset Attribute**

12040 This attribute, of type Structure (Date, Time of Day), represents the date and time at which the *ElapsedActiveTime*  
12041 attribute of a Binary Input, Output or Value cluster was most recently set to a zero value. If one of the optional  
12042 properties *ElapsedActiveTime* or *TimeOfATReset* is present, then both of these attributes shall be present.

### 12043 **9.4.20.11 TimeOfSCReset Attribute**

12044 This attribute, of type Structure (Date, Time of Day), represents the date and time at which the *ChangeOfStateCount*  
12045 attribute of a Binary Input, Output or Value cluster was most recently set to a zero value. If one of the optional  
12046 properties *ChangeOfStateTime*, *ChangeOfStateCount*, or *TimeOfSCReset* is present, then all of these attributes shall  
12047 be present.

### 12048 **9.4.20.12 FeedbackValue Attribute**

12049 This property, of type enumeration, indicates a feedback value from which *PresentValue* must differ before an  
12050 OFFNORMAL event is generated, and to which *PresentValue* must return before a TONORMAL event is generated.  
12051 The manner by which the *FeedbackValue* is determined shall be a local matter.

### 12052 **9.4.20.13 ProfileName Attribute**

12053 This attribute, of type Character string, is the name of a BACnet object profile to which its associated cluster conforms.  
12054 A profile defines a set of additional attributes, behavior, and/or requirements for the cluster beyond those specified  
12055 here.

12056 To ensure uniqueness, a profile name must begin with a vendor identifier code (see Clause 23 of [A1]) in base-10  
12057 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that  
12058 vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that  
12059 publishes and maintains the profile document named by the remainder of the profile name. This vendor identifier need  
12060 not have any relationship to the vendor identifier of the device within which the object resides.

## 12061 **9.4.21 Attributes of BACnet Extended Clusters**

12062 The attributes of BACnet Extended clusters are specifically intended for interworking with BACnet systems (via a  
12063 BACnet gateway). They are based on BACnet properties with the same names. See the BACnet Reference Manual  
12064 [A1] for detailed descriptions of these properties.

12065 References to events and alarms in this section refer to BACnet intrinsic reporting. Note that attribute reporting may  
12066 be used to send reports as well.

### 12067 **9.4.21.1 AckedTransitions Attribute**

12068 This attribute, of type bitmap, holds three one-bit flags (b0, b1, b2) that respectively indicate the receipt of  
12069 acknowledgments for TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events

### 12070 **9.4.21.2 AlarmValue Attribute**

12071 This attribute, of type Boolean, specifies the value that the *PresentValue* attribute must have before a TO-  
12072 OFFNORMAL event is generated.

### 12073 **9.4.21.3 AlarmValues Attribute**

12074 This attribute, of type Set of uint16, specifies any values that the *PresentValue* attribute must equal before a TO-  
12075 OFFNORMAL event is generated.

### 12076 **9.4.21.4 FaultValues Attribute**

12077 This attribute, of type Set of uint16, specifies any values that the *PresentValue* attribute must equal before a TO-  
12078 FAULT event is generated.

### 12079 **9.4.21.5 NotificationClass Attribute**

12080 This attribute, of type uint16, specifies the notification class to be used when handling and generating event  
12081 notifications for this object (over a BACnet gateway).

### 12082 **9.4.21.6 Deadband Attribute**

12083 This attribute, of type single, specifies a range (from  $LowLimit + Deadband$  to  $HighLimit - Deadband$ ) which the  
12084 *PresentValue* must return within for a TO-NORMAL event to be generated.

### 12085 **9.4.21.7 EventEnable Attribute**

12086 This attribute, of type bitmap, holds three one-bit flags (b0, b1, b2) that respectively enable (1) and disable (0)  
12087 reporting of TO-OFFNORMAL, TO-FAULT, and TO-NORMAL events.

### 12088 **9.4.21.8 EventState Attribute**

12089 The *EventState* attribute, of type 8-bit enumeration, is included in order to provide a way to determine if this object  
12090 has an active event state associated with it. The allowed values are:

- 12091 • NORMAL (0)
- 12092 • FAULT (1)
- 12093 • OFFNORMAL (2)
- 12094 • HIGH-LIMIT (3)
- 12095 • LOW-LIMIT (4)

### 12096 **9.4.21.9 HighLimit Attribute**

12097 This attribute, of type single, specifies a limit that *PresentValue* must exceed before an OFF-NORMAL (HIGH-  
12098 LIMIT) event is generated.

12099 **9.4.21.10 LimitEnable Attribute**

12100 This attribute, of type map8, holds two one-bit flags. The flag in bit position 0 enables reporting of low limit off-  
 12101 normal and return-to-normal events if it has the value 1, and disables reporting of these events if it has the value 0.  
 12102 The flag in bit position 1 enables reporting of high limit off-normal and return-to-normal events if it has the value 1,  
 12103 and disables reporting of these events if it has the value 0.

12104 **9.4.21.11 LowLimit Attribute**

12105 This attribute, of type single, shall specify a limit that *PresentValue* must fall below before an OFF-NORMAL (LOW-  
 12106 LIMIT) event is generated.

12107 **9.4.21.12 NotifyType Attribute**

12108 This attribute, of type enumeration, indicates whether the notifications generated by the cluster should be Events (0)  
 12109 or Alarms (1).

12110 **9.4.21.13 TimeDelay Attribute**

12111 This attribute, of type Unsigned 8-bit integer, specifies the minimum period of time in seconds that *PresentValue* must  
 12112 remain outside the band defined by the *HighLimit* and *LowLimit* attributes before a TO-OFFNORMAL event is  
 12113 generated, or within the band (from *LowLimit* + *Deadband* to *HighLimit* - *Deadband*) before a TO-NORMAL event  
 12114 is generated.

12115 **9.4.21.14 EventTimeStamps Attribute**

12116 This optional read-only attribute is of type Array[3]. The three elements each have a type which is one of:

- 12117 • 16-bit unsigned integer - a sequence number
- 12118 • Time of day
- 12119 • Structure of (date, time of day)

12120 The elements of the array hold the times (or sequence numbers) of the last event notifications for TO-OFFNORMAL,  
 12121 TO-FAULT, and TO-NORMAL events, respectively. The type of the elements is discovered by reading the attribute.

12122 **9.5 ISO 7818 Protocol Tunnel**

---

12123 **9.5.1 Scope and Purpose**

12124 This section specifies a single cluster, the ISO7816 Tunnel cluster, which provides commands and attributes for mobile  
 12125 office solutions.

12126 This cluster is to provide a standardized interface to enable a scenario of authorization management on mobile office  
 12127 devices (e.g., access to PC resources)

12128 **9.5.2 Definitions**

12129 The definitions used in the ISO 7816 Protocol Tunnel are shown in Table 9-24.

12130 **Table 9-24. Definitions Used in ISO 7816 Protocol Tunnel Description**

Term	Definition
------	------------

Target Device	A Computer System on which User has to perform authentication in order to access to information services
User Token	A device used by a Target Device to authenticate and authorize User
Virtual SmartCard	A SmartCard that is a node on the network

### 12131 9.5.3 General Description

12132 The cluster specified in this document is typically used for telecom applications, but may be used in any other  
12133 application domains.

### 12134 9.5.4 Overview

12135 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
12136 etc.

12137 This cluster provides attributes and commands to tunnel ISO7816 APDUs, enabling solution such as Mobile Office,  
12138 i.e., a mechanism to authenticate and authorize Users on shared Computer System (said Target Device) by means of  
12139 a Virtual Smartcard (generically said User Token).

12140 A Target Device, enabled by the server side of this cluster, and a User Token (supporting a client side of this cluster)  
12141 can establish a connection and exchange information by means of ISO7816 APDU messages over network.

#### 12142 9.5.4.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

#### 12143 9.5.4.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	T7816	Type 1 (client to server)

#### 12144 9.5.4.3 Cluster Identifiers

Identifier	Name
0x0615	ISO 7818 Protocol Tunnel

### 12145 9.5.5 Server

#### 12146 9.5.5.1 Dependencies

12147 Since ISO7816 protocol may use APDU frames larger than typical payload, stack fragmentation or Partition cluster  
12148 shall be supported by the devices supporting this cluster.

#### 12149 9.5.5.2 Attributes

12150 The ISO7816 Tunnel cluster contains the attribute shown in Table 9-25.

12151

**Table 9-25. Attributes for the ISO7816 Tunnel Cluster**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Default</b>	<b>M/O</b>
0x0001	<i>Status</i>	uint8	0x00-0x01	R	0x00	M

12152 **9.5.5.2.1 Status Attribute**

12153 The *Status* attribute specifies the Server internal state.

12154 Values and usage of this attribute are application dependent, e.g., server busy (client connected). Server supports only  
 12155 one client connection at a time.

12156 The *Status* values are shown in Table 9-26.

12157

**Table 9-26. Status Values**

<b>Meaning</b>	<b>Values</b>
0x00	FREE
0x01	BUSY

12158 **9.5.5.3 Commands Received**

12159 The cluster specific commands received by the ISO7816 Tunnel server cluster are listed in Table 9-27.

12160

**Table 9-27. Received Command IDs for the ISO7816 Tunnel Cluster**

<b>Command Identifier Field Value</b>	<b>Description</b>	<b>M/O</b>
0x00	Transfer APDU	M
0x01	Insert SmartCard	M
0x02	Extract SmartCard	M

12161 **9.5.5.3.1 Transfer APDU Command**

12162 **9.5.5.3.1.1 Payload Format**

12163 The Transfer APDU command shall be formatted as illustrated in Figure 9-5.

12164

**Figure 9-5. Format of the Transfer APDU command**

<b>Bits</b>	Variable
<b>Data Type</b>	octstr
<b>Field Name</b>	APDU

12165 **9.5.5.3.1.2 APDU Field**

12166 The APDU field is of variable length and is an ISO7816 APDU as defined in the ISO7816 standard [I2]

12167 **9.5.5.3.1.3 When Generated**

12168 This command is generated when an ISO7816 APDU has to be transferred across a tunnel.

12169 **9.5.5.3.1.4 Effect on Receipt**

12170 On receipt of this command, a device shall process the ISO7816 APDU as specified in the ISO7816 standard.

12171 **9.5.5.3.2 Insert Smart Card**12172 **9.5.5.3.2.1 Payload Format**

12173 No payload needed for Insert Smart Card command.

12174 **9.5.5.3.2.2 When Generated**

12175 This command is generated when a User Token insertion has to be sent to Server.

12176 **9.5.5.3.2.3 Effect on Receipt**

12177 On receipt of this command:

- 12178 • If the *Status* attribute is equal to BUSY, the Server shall send a Default Response with status FAILURE.
- 12179 • If the *Status* attribute is equal to FREE and the bit ‘Disable Default Response’ of the Frame control field of the  
12180 ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It also shall set its Status  
12181 Attributes to BUSY and it can start to exchange APDUs with Client over ISO7816 Tunnel.

12182 **9.5.5.3.3 Extract Smart Card**12183 **9.5.5.3.3.1 Payload Format**

12184 No payload needed for Insert Smart Card command.

12185 **9.5.5.3.3.2 When Generated**

12186 This command is generated when a User Token extraction has to be sent to Server.

12187 **9.5.5.3.3.3 Effect on Receipt**

12188 On receipt of this command:

- 12189 • If the *Status* attribute is equal to FREE, the Server shall send a Default Response with status FAILURE.
- 12190 • If the *Status* attribute is equal to BUSY and the bit ‘Disable Default Response’ of the of the Frame control  
12191 field of the ZCL Header is set to zero, the Server shall send respond with status SUCCESS. It shall also set its  
12192 Status Attributes to FREE and after this, Server shall not be able to exchange APDUs with Client over  
12193 ISO7816 Tunnel.

12194 **9.5.5.4 Commands Generated**

12195 The cluster specific commands generated by the ISO7816 Tunnel server cluster are listed in Table 9-28.

12196 **Table 9-28. Generated Command IDs for the ISO7816 Tunnel Cluster**

Command Identifier Field Value	Description	M/O
0x00	Transfer APDU	M

12197 **9.5.5.5 Transfer APDU**12198 **9.5.5.5.1.1 Payload Format**



12199 The Transfer APDU command shall be formatted using the same command “Transfer APDU” in paragraph 9.5.5.3.1.  
12200 The effect on receipt is the same as reported in 9.5.5.3.1.4.

## 12201 **9.5.6 Client**

### 12202 **9.5.6.1 Dependencies**

12203 None

### 12204 **9.5.6.2 Attributes**

12205 The client cluster has no attributes.

### 12206 **9.5.6.3 Command Received**

12207 The client receives the cluster specific commands detailed in 9.5.5.4 as required by application profiles

### 12208 **9.5.6.4 Command Generated**

12209 The client generates the cluster specific commands detailed in 9.5.5.3 as required by application profiles.

## 12210 **9.6 Partition**

---

### 12211 **9.6.1 Scope and Purpose**

12212 This section specifies a single cluster, the Partition cluster, which provides commands and attributes for enabling  
12213 partitioning of large frame to be carried from other clusters. This cluster is designed to provide a standardized interface  
12214 for the applications to manage extended size frame format, up to 100KB long. The Partition cluster can be used in  
12215 different application scenarios that requires extended frame for services provided by particular clusters.

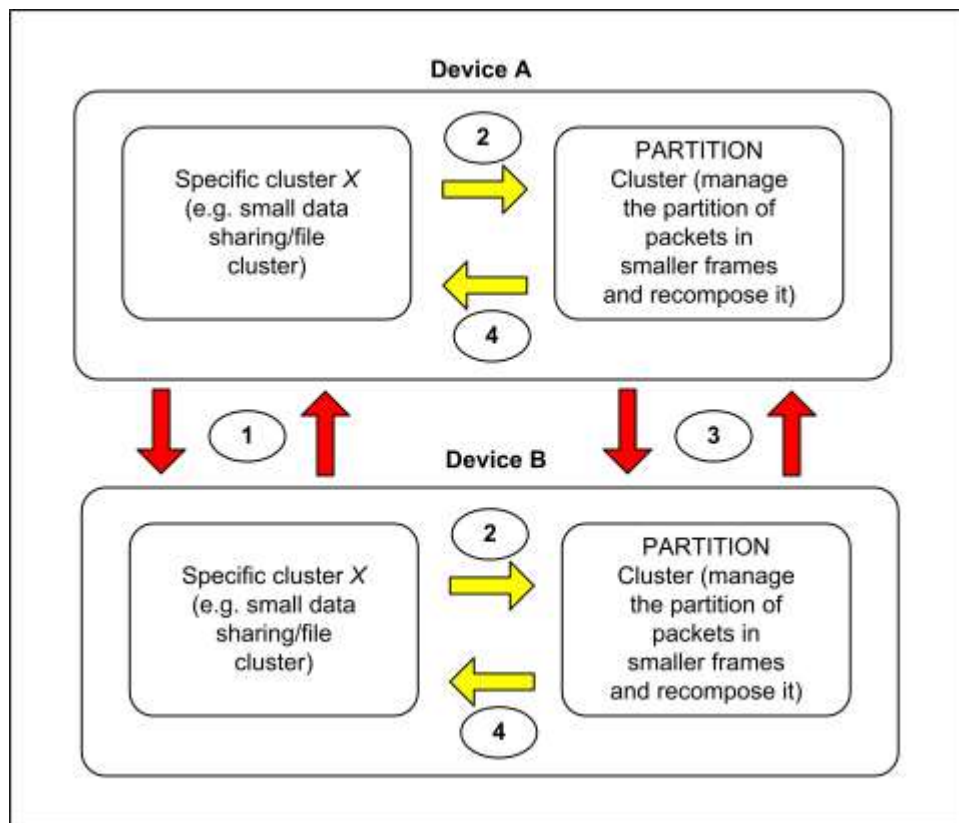
### 12216 **9.6.2 Introduction**

12217 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
12218 etc.

12219 The cluster specified in this may be used in different application domains. The Partition cluster provides the attributes  
12220 and commands required for enabling and managing the transmission of extended frames over a network.

12221

Figure 9-6. Typical Usage of the Partition Cluster



12222

12223

12224 The typical usage of Partition cluster is shown in Figure 9-6 and can be represented by the following phases:

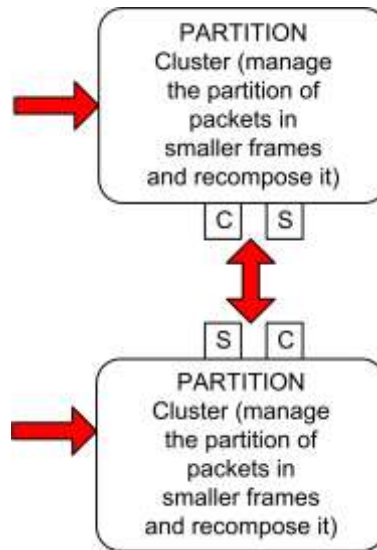
- 12225 3. Cluster based Discovery (e.g., performing Match\_Desc\_Req) can be operated to the specific cluster X that  
12226 needs to transfer information to a matching cluster (e.g., File Cluster); moreover cluster based discovery should  
12227 be used in order to check the support of the Partition Cluster by a recipient device.
- 12228 4. If the application entity requires transmission of large frames (e.g., an application willing to use data  
12229 sharing/file cluster, generic tunnel cluster) the specific application entity shall subscribe to the Partition cluster;  
12230 registration or subscription phase is described in 9.6.5.
- 12231 5. The Partition clusters will perform and manage the “fragmentation” and send the rebuilt frame to the registered  
12232 specific cluster.
- 12233 6. The Partition Cluster will forward the recomposed packet to the specific clusters that registered to the  
12234 Partitioning Cluster (e.g., Cluster X).

12235 The application object implementing and using the Partition Cluster should have enough memory to manage the  
12236 incoming frames; the Partition cluster is designed for devices like Mobile Phones or other gateways that have extended  
12237 computing capabilities in comparison with typical devices.

12238 Since the Partition cluster performs a handshake phase between the devices using Partition cluster (reading and writing  
12239 the proper defined attributes) as described with more details in 9.6.5, both client and server should be used in order to  
12240 guarantee a full bidirectional link in the communication (see Figure 9-7).

12241

Figure 9-7. Client and Server in Partition Cluster



12242

12243

12244 A simple way to enable the use of the partition cluster should be to define a specific API that would support the  
 12245 sending/receive functionalities through the use of *Partition Cluster*. Partition should be considered like a specific  
 12246 tunnel cluster: Commands exposed to the application objects (general API to be used by the application) should be  
 12247 the following ones:

- 12248 • *TransferFrameUsingPartitionCluster* (send/receive) → the max size for the carried data is typically  
 12249 25KB<x<100KB as from discussed requirements. This command may pass a handler to the sequence of bytes  
 12250 corresponding to the ZCL message of the specific cluster using the Partition Cluster. In order to operate using  
 12251 the Partition Cluster the application may want to manage the transmission and reception of large frames  
 12252 running the handshake phase described in 9.6.5.

12253 Rather than pushing the large frame to the application, the Partition Cluster may only inform the application  
 12254 that a packet has arrived (very short packet that can be fed through the stack). The application will then read the  
 12255 frame from the Partitioning Cluster. The detailed mechanism to perform this operation is out of scope of this  
 12256 specification

- 12257 • *RW handshake commands*

12258 Partition cluster related commands should be sent transparently between the application objects managing the  
 12259 fragmentation to guarantee the reconstruction of the received frame; these commands are described in the following  
 12260 sections:

- 12261 • *Transfer partitioned frame* (max dimension<max size carried by the ZCL standard frame ~80B)
- 12262 • *Multiple ACKs*

12263 In the Partition Cluster attributes a list of registered clusters should be inserted in order to manage possible sharing  
 12264 and re-use of it by multiple clusters.

### 12265 9.6.2.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

12266 **9.6.2.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	PART	Type 1 (client to server)

12267 **9.6.2.3 Cluster Identifiers**

Identifier	Name
0x0016	Partition

12268 **9.6.3 Server**

12269 **9.6.3.1 Dependencies**

12270 None

12271 **9.6.3.2 Attributes**

12272 The attributes are used in the Partition Cluster summarized in Table 9-29.

12273 **Table 9-29. Attributes of the Partition Cluster**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>MaximumIncomingTransferSize</i>	uint16	0x0000-0xffff	R	0x0500	M
0x0001	<i>MaximumOutgoingTransferSize</i>	uint16	0x0000-0xffff	R	0x0500	M
0x0002	<i>PartitionedFrameSize</i>	uint8	0x00-0xff	RW	0x50	M
0x0003	<i>LargeFrameSize</i>	uint16	0x0000-0xffff	RW	0x0500	M
0x0004	<i>NumberOfACKFrame</i>	uint8	0x00-0xff	RW	0x64	M
0x0005	<i>NACKTimeout</i>	uint16	0x0000-0xffff	R	<i>apsAckWait Duration + InterframeDelay * NumberOfACK Frames</i>	M
0x0006	<i>InterframeDelay</i>	uint8	Default-0xff	RW	<i>apsInterFrame Delay</i>	M
0x0007	<i>NumberOfSendRetries</i>	uint8	0x00-0xff	R	0x03	M
0x0008	<i>SenderTimeout</i>	uint16	Default-0xffff	R	<i>2*apsAckWait Duration + InterframeDelay * NumberOfACK Frames</i>	M

Id	Name	Type	Range	Acc	Default	M/O
0x0009	<i>ReceiverTimeout</i>	uint16	Default-0xffff	R	<i>apsAckWait</i> <i>Duration</i> + <i>InterframeDelay</i> + <i>NumberOfSendRetries</i> * <i>NACKTimeout</i>	M

12274 **9.6.3.2.1.1 MaximumIncomingTransferSize Attribute**

12275 The *MaximumIncomingTransferSize* attribute specifies the maximum size, as multiple of *PartitionedFrameSize*, of the  
 12276 application service data unit (ASDU) that can be transferred to this node in one single message transfer. The ASDU  
 12277 referred to is the ZCL frame, including header and payload, of any command received by a Partition cluster on the  
 12278 same endpoint.

12279 **9.6.3.2.1.2 MaximumOutgoingTransferSize Attribute**

12280 The *MaximumOutgoingTransferSize* attribute specifies the maximum size, as multiple of *PartitionedFrameSize*, of the  
 12281 application service data unit (ASDU) that can be transferred from this node in one single message transfer. The ASDU  
 12282 referred to is the ZCL frame, including header and payload, of any command received by a Partition cluster on the  
 12283 same endpoint.

12284 **9.6.3.2.1.3 PartitionedFrameSize Attribute**

12285 The *PartitionedFrameSize* attribute specifies the size in bytes of a partitioned frame transferred using  
 12286 *TransferPartitionedFrame* command. The default value for this attribute is equal to 80 bytes (0x50) because a “large  
 12287 frame” to be transferred using the Partition Cluster shall be partitioned into smaller *PartitionedFrameSize* frame size.

12288 **9.6.3.2.1.4 LargeFrameSize Attribute**

12289 The *LargeFrameSize* attribute specifies the size, in multiple of *PartitionedFrameSize*, of a large frame to be partitioned  
 12290 using the Partition cluster into *PartitionedFrameSize* bytes carried by *TransferPartitionedFrame* commands. The  
 12291 default value of this attribute should be set equal to 0x0500 (so that, given the default *PartitionedFrameSize* attribute  
 12292 equal to 80bytes the default large frame would be 100KB). The length in byte of the large frame to be partitioned is  
 12293 equal to *PartitionedFrameSize\*LargeFrameSize*. In case the frame to be partitioned is not multiple of  
 12294 *PartitionedFrameSize\*LargeFrameSize*, the last *TransferPartitionedFrame* command shall be padded with zeros in  
 12295 order to fit in *PartitionedFrameSize* length of the last *TransferPartitionedFrame* command.

12296 **9.6.3.2.1.5 NumberOfACKFrame Attribute**

12297 The *NumberOfACKFrame* attribute specifies the number of partitioned frames to be received before sending a multiple  
 12298 acknowledge command. The proper setting of this attribute guarantee the reduction of acknowledge packet to be  
 12299 transmitted over the network. If *NumberOfAckFrame* attribute is set to 0x00, it indicates a non-ACK transmission. In  
 12300 this case, the sender would ignore the sender timeout and send the blocks continuously with *InterframeDelay* interval  
 12301 between each partitioned frame. In this case the receiver shall not return the *MultipleACK* after receiving the block,  
 12302 and the *ReceiverTimeout* and *NACKTimeout* attributes (set to the receiver) shall be also ignored.

12303 **9.6.3.2.1.6 NACKTimeout Attribute**

12304 *NACKTimeout* attribute specifies the maximum time, expressed in milliseconds, the receiver entity should wait after  
 12305 having received the last *NumberOfAckFrame* partitioned frames, before sending a *MultipleACK* command to the  
 12306 sender. The receiver shall transmit immediately if it receives all the partitioned frames correctly.

12307 **9.6.3.2.1.7 InterFrameDelay Attribute**

12308 The *InterFrameDelay* attribute specifies the delay in milliseconds between successive transmissions of  
 12309 *TransferPartitionedFrame* commands. Default value for this attributes is given by the *apsInterFrameDelay*. 0x00 is  
 12310 not a valid value for this attribute. If the device doesn’t support APS fragmentation but supports the Partition Cluster,  
 12311 this value shall be set to 10ms.

12312 **9.6.3.2.1.8 NumberOfSendRetries Attribute**

12313 The *NumberOfSendRetries* specifies the maximum number of retries the sender should perform in case no  
12314 *MultipleACK* have been received in *SenderTimeout* time period. This attribute should be reset to the default value  
12315 when a *MultipleACK* command is received.

12316 **9.6.3.2.1.9 SenderTimeout Attribute**

12317 The *SenderTimeout* attribute specifies is the time that the sender should wait for the *MultipleACK* before sending a  
12318 number of *NumberOfACKFrame* of *TransferPartitionedFrame* commands again. This attribute should be reset to the  
12319 default value when a *MultipleACK* command is received and started with the first block sent to the receiver.

12320 **9.6.3.2.1.10 ReceiverTimeout Attribute**

12321 The *ReceiverTimeout* attribute specifies the maximum time the receiver need to wait for a *TransferPartitionedFrame*  
12322 command after the reception the first frame of the large frame to be transferred. If there will be no frames received  
12323 after *ReceiverTimeout*, the receiver will exit the Partition procedure.

12324 **9.6.3.3 Commands Received**

12325 The received command IDs for the Partition cluster are listed in Table 9-30.

12326 **Table 9-30. Server Received Command IDs for the Partition Cluster**

Command Identifier Field Value	Description	M/O
0x00	TransferPartitionedFrame	M
0x01	ReadHandshakeParam	M
0x02	WriteHandshakeParam	M

12327 **9.6.3.3.1 TransferPartitionedFrame Command**

12328 The *TransferPartitionedFrame* command is used to send a partitioned frame to another Partition cluster. It shall be  
12329 originated by the sender device and sent to the recipient device which is expected to answer with a *MultipleACK* (as  
12330 defined in 9.6.3.4.1). When the sender composes and sends to the receiver the first *TransferPartitionedFrame*  
12331 command, a timer on the sender is started; this timer shall be used to check if the sender received a *MultipleACK*  
12332 before *SenderTimeout* time period. The sender may wait for a *MultipleACK* after every *NumberOfACKFrame* blocks  
12333 transmission. In that case the value *NumberOfACKFrame* should be set in a handshake phase. The sender will consider  
12334 a successful transmission of a *NumberOfACKFrame* number of blocks if no *NACKIDs* are carried by the *MultipleACK*  
12335 command payload.

12336 The *TransferPartitionedFrame* command shall be formatted as illustrated in Figure 9-8.

12337 **Figure 9-8. Format of the TransferPartitionedFrame Command**

octets	1	1-2	Variable
Data Types	map8	uint8 or uint16	octstr
Field Name	<i>Fragmentation Options</i>	<i>PartitionIndicator</i>	<i>PartitionedFrame</i>

12338 The *Fragmentation Options* field shall be formatted as in Figure 9-9.

12339

**Figure 9-9. Format of the *FragmentationOptions* Field**

<b>b0: 1 bit</b>	<b>b1: 1 bit</b>	<b>b2-b7: 6 bit</b>
<i>First block</i>	<i>Indicator length</i>	Reserved

12340 *First Block* field  $b0=1$  indicates that the *TransferPartitionedFrame* command carries the first block of  
 12341 *NumberOfACKFrame* while  $b0=0$  indicates that the *TransferPartitionedFrame* command doesn't carry a first block.  
 12342 *Indicator length* field specifies if the *PartitionIndicator* field is 1 or 2-bytes long:  $b1=0$  indicates that the  
 12343 *PartitionIndicator* is 1-byte long,  $b1 = 1$  indicates that the *PartitionIndicator* is 2-bytes long.

12344 *PartitionIndicator* field specifies the overall number of blocks for the 1st partitioned frame (fragment), and the block  
 12345 index for the other fragments starting from 0x01 or 0x0001 (respectively for  $b1=0$  or  $b1 = 1$ ).

12346 The address mechanism used for the *TransferPartitionedFrame* command should not use broadcasting and it should  
 12347 not use multicasting.

12348 **9.6.3.3.1.1 Effect on Receipt**

12349 The receiver will start receiving *TransferPartitionedFrame* commands and start the *NACKTimeout* and  
 12350 *ReceiverTimeout* timers after the reception of the first frame related to the transaction registered by the handshake  
 12351 phase (*WriteHandshakeParam* command); if *NumberOfACKFrames* have been received, the Partition Cluster of the  
 12352 receiver will send a *MultipleACK* command with no *NACKId*. The block indexes of expected  
 12353 *TransferPartitionedFrame* commands that have not been received in *NACKTimeout* (*NACKIds*) will be inserted in the  
 12354 *MultipleACK* command returned to the sender. If there are no frames received after *ReceiverTimeout*, the receiver will  
 12355 exit the partition procedure. In case the receiver receives a number equal to *NumberOfACKFrame* partitioned frames  
 12356 it shall send the *MultipleACK* command without waiting for a *NACKTimeout* time. The receiver will also reset the  
 12357 *ReceiverTimeout* timer after reception of a *TransferPartitionedFrame* command.

12358 **9.6.3.3.2 ReadHandshakeParam Command**

12359 The *ReadHandshakeParam* command is used in order to read the appropriate set of parameters for each transaction  
 12360 to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific cluster referred to the  
 12361 large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the specific frame  
 12362 to be partitioned shall be carried directly in the ZCL header.

12363 **Figure 9-10. *ReadHandshakeParam* Frame**

<b>Octets</b>	2	2	...	2
<b>Data Types</b>	ClusterID	AttributeID	...	AttributeID
<b>Field Name</b>	Partitioned ClusterID	Attribute identifier 1	...	Attribute identifier <i>n</i>

12364 **9.6.3.3.3 WriteHandshakeParam Command**

12365 The *WriteHandshakeParam* command is used during the handshake phase in order to write the appropriate parameters  
 12366 for each transaction to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific  
 12367 cluster referred to the frames that is going to be partitioned by the Partition Cluster itself. The transaction number of  
 12368 the specific frame to be partitioned shall be carried in the ZCL header. See 2.4.3for write attribute record format. By  
 12369 using the *WriteHandshakeParam* command report it is possible to write Partition Cluster attributes related to the  
 12370 specific large frame to be transferred using partitioning.

12371

Figure 9-11. *WriteHandshakeParam* Frame

<b>Octets</b>	2	2	...	2
<b>Data Types</b>	ClusterID	See 2.4.3	...	See 2.4.3
<b>Field Name</b>	Partitioned ClusterID	Write Attribute Record 1		Write Attribute Record <i>n</i>

12372

12373

Figure 9-12. Format of Write Attribute Record Field

<b>octets: 2</b>	<b>1</b>	<b>Variable</b>
Attribute Identifier	Attribute Data Type	Attribute Data

### 12374 9.6.3.4 Commands Generated

12375 The generated command IDs for the server Partition cluster are listed in Table 9-31.

12376

Table 9-31. Generated Command IDs for the Partition Cluster

Command Identifier Field Value	Description	M/O
0x00	<i>MultipleACK</i>	M
0x01	<i>ReadHandshakeParamResponse</i>	M

#### 12377 9.6.3.4.1 MultipleACK Command

12378 The receiver shall return the *MultipleACK* command when receiving a number equal to *NumberOfACKFrame*  
 12379 *TransferPartitionedFrame* commands (partitioned frames) or when *NACKTimeout* expires. The *MultipleACK*  
 12380 command will carry no *NACKId* in the payload if *NumberOfACKFrame TransferPartitionedFrame* commands are  
 12381 received. The sender may wait for a *MultipleACK* command after every *NumberOfACKFrame* blocks transmission.  
 12382 The *MultipleACK* command shall be formatted as illustrated in Figure 9-13.

12383

Figure 9-13. Format of the *MultipleACK* Command

<b>Octets</b>	1	1-2	1-2	1-2	1-2
<b>Data Types</b>	map8	uint8 or uint16	uint8 or uint16		uint8 or uint16
<b>Field Name</b>	<i>ACK Options</i>	<i>FirstFrameID</i>	<i>NACKId</i>	...	<i>NACKId</i>

12384

12385 The *ACKOptions* payload fields shall be formatted as illustrated in Figure 9-14.

12386

Figure 9-14. Format of the *ACK Options* Field

<b>b0: 1 bit</b>	<b>b1-b7: 7 bit</b>
<i>NACKId length</i>	Reserved



12387 *NACKId length* specifies if the *NACKId*, corresponding to the *PartitionIndicator* (*NACKIds* carried in this command  
 12388 are the values of the "PartitionIndicator" field), and the *FirstFrameID* are 1 or 2 bytes long: *b0=0* indicates that the  
 12389 *NACKIds* and the *FirstFrameID*, are 1-byte long, *b0 = 1* indicates that the *NACKIds* and the *FirstFrameID*, are 2-  
 12390 bytes long.

12391 *FirstFrameID* field indicates the first partition frame (block) index of the current overall *NumberOfACKFrame* blocks  
 12392 the *MultipleACK* refers to. It is used in order to identify the set of *NumberOfACKFrame* the *MultipleACK* command  
 12393 refers to.

12394 *NACKId* fields represent the ID of partitioned frame that have not been received yet after *NACKTimeout*.

12395 **9.6.3.4.1.1 Effect on Receipt**

12396 After sending a number of *TransferPartitionedFrame* commands equal to *NumberOfACKFrame* (Number of  
 12397 acknowledged frames) the sender will wait for a *MultipleACK*: a successful transmission is indicated by a  
 12398 *MultipleACK* command with no *NACKId* fields carried. The sender shall stop sending the next *NumberOfACKFrame*  
 12399 blocks until it receives a *MultipleACK* command reporting a successful transmission.

12400 When the sender successfully sends the current *NumberOfACKFrame* blocks and receives a *MultipleACK* command  
 12401 with no *NACKId* fields, the Partition Cluster should proceed to send the next *NumberOfACKFrame* set of blocks of  
 12402 the, large frame to be transmitted, until all the set of blocks have been sent out. The partition parameters such as  
 12403 *NumberOfACKFrame* may be tuned after sending out the current *NumberOfACKFrame*, set of blocks (e.g., the value  
 12404 of *NumberOfACKFrame* may be decreased after retransmissions of many *TransferPartitionedFrame* commands of a  
 12405 previous transaction).

12406 In case the receiver does need to send out several *MultipleACKs* to the sender, it should not send out a next one until  
 12407 completing the reception of all blocks indicated in the *NACKId* fields of the previous *MultipleACK*. The sender should  
 12408 receive *MultipleACK* command by sender timeout (this timeout specifies how long to wait for a *MultipleACK*); if no  
 12409 *MultipleACK* command is received the sender will retransmit the *TransferPartitionedFrame* commands up to a  
 12410 maximum number of retries (in order to optimize the protocol the sender may reduce also the *NumberOfACKFrame*  
 12411 value by using the writing command defined in the handshake phase); if the sender doesn't receive any *MultipleACK*  
 12412 after maximum number of retries it will exit the partition procedure and the *TransferFrameUsingPartitionCluster*  
 12413 response will notify the error in the partition procedure; otherwise, if *MultipleACK* is received carrying some *NACK*  
 12414 *IDs*, the sender will reset the sender timeout and the max number of retries and resend the no acknowledged  
 12415 *TransferPartitionedFrame* commands up to max number of retries until a *MultipleACK* with no *NACK* is received  
 12416 (success in the partition transaction) or the *SenderTimeout* expires (in case no *MultipleACK* commands are received)  
 12417 or max number of retries reached (in case *MultipleACK* commands are received but still with *NACKIDs*).

12418 The *SenderTimeout* is equal to  $2 * apcAckWaitDuration + InterframeDelay * NumberOfACKFrames$ .

12419 **9.6.3.4.2 ReadHandshakeParamResponse Command**

12420 The *ReadHandshakeParamResponse* command is used in order to response to the corresponding  
 12421 *ReadHandshakeParam* command in order to communicate the appropriate set of parameters configured for each  
 12422 transaction to be performed by the Partition Cluster. The *Partitioned ClusterID* field identifies the specific cluster  
 12423 referred to the large frame that is going to be partitioned by the Partition Cluster itself. The transaction number of the  
 12424 specific frame to be partitioned shall be carried directly in the ZCL header. The rRead Attribute status record field is  
 12425 the same as defined for the ZCL (see 2.4.2.1 ).

12426 **Figure 9-15. ReadHandshakeParamResponse Frame**

<b>octets</b>	2	Variable	...	Variable
<b>Data Types</b>	ClusterID	See 2.4.2.1	...	See 2.4.2.1
<b>Field Name</b>	Partitioned ClusterID	Read attribute status record 1	...	Read attribute status record <i>n</i>

12427

12428

**Figure 9-16. Format of Read Attribute Status Record Field**

octets: 2	2	0/1	0/Variable
Attribute Identifier	Status	Attribute Data Type	Attribute Data

12429 **9.6.4 Client**12430 **9.6.4.1 Attributes**

12431 None

12432 **9.6.4.2 Command Received**

12433 The client receives the cluster specific commands detailed in 9.6.3.4, as required by application profiles.

12434 **9.6.4.3 Command Generated**

12435 The client generates the cluster specific commands detailed in 9.6.3.3 as required by application profiles.

12436 **9.6.5 General Use of Partition Cluster**

12437 The Partition cluster may be used by multiple clusters defined in a single application object. In order to perform the  
 12438 recognition of multiple partitioned frames associated to a specific cluster and reconstruct a partitioned large frame the  
 12439 Partition cluster shall maintain an internal table similar to the one presented in Table 9-32: Each large frame to be  
 12440 partitioned can be identified by the ClusterID and the ZCL transaction sequence number.

12441 The specific clusters using the Partition cluster to transfer large frame shall subscribe to the registration table writing  
 12442 an entry for each frame to be partitioned with the Partition Cluster attributes fields specified in 9.6.3.2. This entry shall  
 12443 be cancelled by the Partition Cluster when the frame is correctly transferred or the partitioning procedure exited with  
 12444 errors. The entries of this table should be inserted during the handshake phase, i.e., in the sender when the  
 12445 *WriteHandshakeParam* command is generated and in the receiver when the *WriteHandshakeParam* command is  
 12446 received.

12447 The partitioned frames generated from the partitioning of a large frame shall use the ZCL transaction sequence number  
 12448 inserted in the ZCL header of the large frame for each small partitioned frame (transferred using the  
 12449 *TransferPartitionedFrame* commands) in order to identify the proper fragment if multiple partitions are running on  
 12450 the same endpoint with large frames carrying the same ClusterIDs.

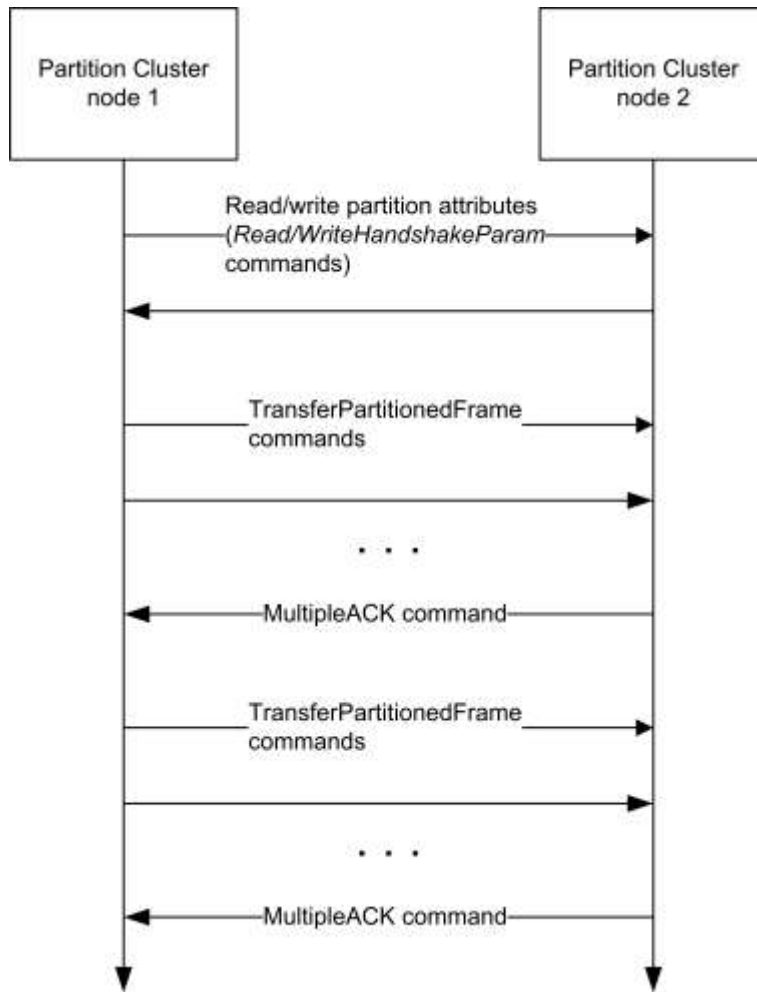
12451 **Table 9-32. Registration Table of Clusters Using the Partition Cluster**

ClusterID	Transaction sequence number	<i>Partition Cluster attributes</i>
Registered cluster ID	Transaction sequence number (of the packet to be partitioned) through the Partition cluster	Attributes that are written using the <i>WriteHandshakeParam</i> command

12452

12453

Figure 9-17. Example of Partition Cluster Use



12454

12455

12456

## 9.7 11073 Protocol Tunnel

12457

### 9.7.1 Overview

12458

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

12459

12460

The 11073 Protocol Tunnel cluster provides the commands and attributes required to tunnel the 11073 protocol. The server cluster receives 11073 APDUs and the client cluster generates 11073 APDUs, thus it is necessary to have both server and client on an endpoint to tunnel 11073 messages in both directions.

12461

12462

12463

Commands and attributes are provided for establishing, querying the status of, and removing an 11073 tunnel connection between two devices.

12464

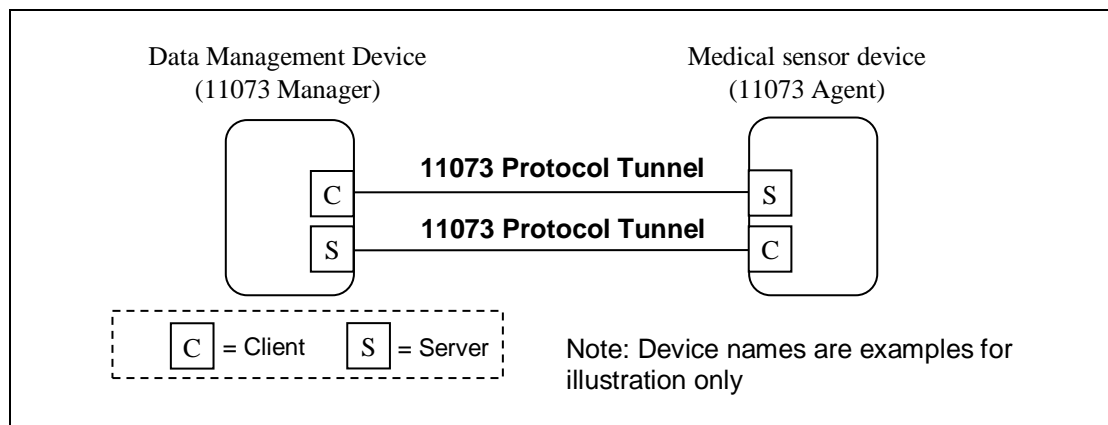
12465

Devices that support this cluster shall also comply with the ISO/IEEE 11073-20601 standard for Personal Health Device Communication [H1] and the applicable ISO/IEEE 11073 device specialization documents [H2] – [H12].

12466

12467

Typical usage of the 11073 Protocol Tunnel cluster is illustrated in Figure 9-18.



12468 **Figure 9-18 Typical Usage of the 11073 Protocol Tunnel cluster**

12469 Note that all 11073 protocol tunnel cluster specific commands are generated by the client and received by the server.  
12470 A typical sequence of events to initiate an 11073 interaction might be:

- 12471 - DMD transmits Connect Request command to sensor (client→server)
- 12472 - Sensor responds with a Connect Status Notification command with status CONNECTED (client→server)
- 12473 - Sensor and DMD carry out an 11073 layer interaction by exchanging Transfer APDU commands in each direction  
12474 (client→server in each case)

### 12475 9.7.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 12476 9.7.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	T11073	Type 1 (client to server)

### 12477 9.7.1.3 Cluster Identifiers

Identifier	Name
0x0614	11073 Protocol Tunnel

## 12478 9.7.2 Server

### 12479 9.7.2.1 Dependencies

12480 Any endpoint that supports the 11073 Protocol Tunnel server cluster shall also support the Generic Tunnel server  
12481 cluster (see 9.2).

12482 The value of the *ProtocolAddress* attribute of the associated Generic Tunnel server cluster shall be set equal to the  
12483 system ID of the 11073 device represented on that endpoint (see [H1]). The system ID, represented as a octet string,  
12484 shall be 8 octets in Big Endian order.

12485 The value of the *MaximumIncomingTransferSize* attribute and the value of the *MaximumOutgoingTransferSize*  
 12486 attribute of the associated Generic Tunnel server cluster shall be set equal to or greater than the maximum APDU size  
 12487 specified in the applicable ISO/IEEE 11073 device specialization document ([H2] – [H12]).

### 9.7.2.2 Attributes

12488  
 12489 The 11073 Protocol Tunnel server cluster contains the attributes shown in Table 9-33:

Table 9-33 – Attributes of the 11073 Protocol Tunnel server cluster

ID	Name	Type	Range	Acc	Default	M/O
0x0000	<i>DeviceIDList</i>	array of uint16	Any valid	R	0xffff	O
0x0001	<i>ManagerTarget</i>	IEEE address	Any valid IEEE address	R	-	O
0x0002	<i>ManagerEndpoint</i>	uint8	0x01-0xff	R	-	O
0x0003	<i>Connected</i>	bool	TRUE / FALSE	R	FALSE	O
0x0004	<i>Preemptible</i>	bool	TRUE / FALSE	R	TRUE	O
0x0005	<i>IdleTimeout</i>	uint16	0x0001 – 0xffff	R	0x0000	O

12491  
 12492 Although the *ManagerTarget*, *ManagerEndpoint*, *Connected*, *Preemptible* and *IdleTimeout* attributes are listed above  
 12493 as optional, if any one of them is implemented then all of them shall be implemented, and also reception of the connect  
 12494 and disconnect request commands shall be implemented, and the 11073 Protocol Tunnel client cluster implemented  
 12495 on the same endpoint shall implement transmission of the connect status notification command.

#### 9.7.2.2.1 DeviceIDList attribute

12496  
 12497 The *DeviceIDList* attribute specifies all devices supported behind a single instance of the 11073 tunnel, on a single  
 12498 endpoint. It allows for discovering the functionality of 11073 devices, e.g. prior to establishment of an 11073 tunnel.  
 12499 The *DeviceIDList* attribute can be read using generic ZCL commands.

12500 For a multifunction device as defined in [Z6], the *DeviceIDList* attribute is mandatory and shall contain a complete  
 12501 list of supported Device IDs (as defined in [Z6]) supported by the single instance of the *11073 Protocol Tunnel* on  
 12502 this particular endpoint. The DeviceID contained in the Simple Descriptor of the multifunction sensor shall contain  
 12503 the value corresponding to the multifunction device itself (see [Z6]).

12504 For all other devices types, the *DeviceIDList* attribute is optional. If implemented, it shall contain the single DeviceID  
 12505 allocated to that device (see [Z6]).

#### 9.7.2.2.2 ManagerTarget attribute

12506  
 12507 The *ManagerTarget* attribute specifies the IEEE address of the currently or most recently connected Data Management  
 12508 device.

12509 **9.7.2.2.3 ManagerEndpoint attribute**

12510 The *ManagerEndpoint* attribute specifies the endpoint used by the currently or most recently connected Data  
12511 Management device.

12512 **9.7.2.2.4 Connected attribute**

12513 The *Connected* attribute specifies whether or not the 11073 tunnel on this endpoint is currently connected.

12514 If this attribute takes the value TRUE, then the tunnel is currently connected.

12515 If this attribute takes the value FALSE, then the tunnel is not currently connected.

12516 Whenever the value of this attribute changes the 11073 layer shall be informed via the transport connected and  
12517 transport disconnected indications.

12518 **9.7.2.2.5 Preemptible attribute**

12519 The *Preemptible* attribute specifies whether or not the current connection can be disconnected by a Data Management  
12520 device other than by the one currently connected.

12521 If this attribute takes the value TRUE, then a disconnect request from a device other than the Data Management device  
12522 indicated by the *ManagerTarget* attribute shall be accepted and if the 11073 tunnel on this endpoint is currently  
12523 connected then it shall become disconnected, and a connect status notification command with status  
12524 DISCONNECTED shall be sent to the currently connected Data Management device.

12525 If this attribute takes the value FALSE, then a disconnect request from a device other than the Data Management  
12526 device indicated by the *ManagerTarget* attribute shall be rejected and a connect status notification command with  
12527 status NOT\_AUTHORIZED sent to the requester.

12528 **9.7.2.2.6 Idle timeout attribute**

12529 The *Idle Timeout* attribute specifies the inactivity time in minutes which the Data Management device will wait  
12530 without transmitting or receiving any tunneled frames to or from the connected target, before it disconnects the  
12531 connection.

12532 If the Data Management device does not intend to timeout this connection after a specific idle period then this attribute  
12533 shall take the value 0xffff.

12534 If the indicated timeout period passes with no data on the 11073 tunnel, then the agent device shall set its *Connected*  
12535 attribute to FALSE and a connect status notification command with status DISCONNECTED shall be sent to the  
12536 currently connected Data Management device. In order to continue to use the tunnel, the agent device shall send the  
12537 Data Management device a further connect status notification command with status RECONNECT\_REQUEST, and  
12538 wait for the Data Management device to respond.

12539 **9.7.2.3 Commands Received**

12540 The cluster specific commands received by the 11073 Protocol Tunnel server cluster are listed in Table 9-34:

12541

12542 **Table 9-34 – Command IDs for the 11073 protocol tunnel cluster**

Command identifier field value	Description	Mandatory/Optional
0x00	Transfer APDU	M
0x01	Connect request	O

Command identifier field value	Description	Mandatory/Optional
0x02	Disconnect request	O
0x03	Connect status notification	O

12543

12544 Although the connect request and disconnect commands are listed above as optional, if reception of either of them is  
 12545 implemented then reception of both of them shall be implemented, and also the *ManagerTarget*, *ManagerEndpoint*,  
 12546 *Connected*, *Preemptible* and *IdleTimeout* attributes shall be implemented, and the 11073 Protocol Tunnel client cluster  
 12547 implemented on the same endpoint shall implement transmission of the connect status notification command.

12548 Although reception of the connect status notification command is listed above as optional, if reception of this  
 12549 command is implemented then also the connect request command shall be implemented by the 11073 Protocol Tunnel  
 12550 server on the same endpoint.

12551 **9.7.2.3.1 Transfer APDU Command**

12552 The Transfer APDU command payload shall be formatted as illustrated in Figure 9-19:

<b>Bits</b>	Variable
<b>Data Type</b>	long octet string
<b>Field Name</b>	APDU

12553 **Figure 9-19 – Transfer APDU payload**

12554 The APDU field is of variable length and is a 11073 APDU as defined in the ISO/IEEE 11073 standard [H1].

12555 **9.7.2.3.1.1 When generated**

12556 This command is generated when an 11073 network layer wishes to transfer an 11073 APDU across a tunnel to another  
 12557 11073 network layer.

12558 The most stringent reliability characteristic of a given transport technology is “Best” reliability. Note - For ZigBee,  
 12559 this corresponds to use of APS-ACKs.

12560 The least stringent reliability characteristic of a given transport technology is “Good” reliability. Note - For ZigBee,  
 12561 this corresponds to no use of APS-ACKs.

12562 The application is responsible for transmitting at a reliability level appropriate for each frame.

12563 This command shall always be transmitted with the disable default response bit in the ZCL frame control field set to  
 12564 1.

12565 **9.7.2.3.1.2 Effect on Receipt**

12566 On receipt of this command, a device shall process the 11073 APDU as specified in [H1] and the applicable device  
 12567 specialization [H2] to [H12]

12568 **9.7.2.3.2 Connect Request Command**

12569 The Connect Request command payload shall be formatted as illustrated in Figure 2.

<b>Octets</b>	1	2	8	1
<b>Data Type</b>	map8	uint16	IEEE address	uint8
<b>Field Name</b>	Connect control	Idle timeout	Manager target	Manager endpoint

Figure 9-20 – Connect Request command payload

12570

12571 **9.7.2.3.2.1 Connect control**

12572 The *connect control* field shall be formatted as illustrated in Figure 9-21:

<b>Bit</b>	0	1-7
<b>Field Name</b>	Preemptible	Reserved

Figure 9-21 – Connect control field format

12573

12574 The *Preemptible* bit shall indicate whether or not this connection can be removed by a different Data Management  
12575 device.

12576 **9.7.2.3.2.2 Idle timeout**

12577 The *idle timeout* field shall indicate the inactivity time in minutes which the Data Management device will wait without  
12578 receiving any tunneled frames from the connected target, before it disconnects the connection.

12579 **9.7.2.3.2.3 Manager target**

12580 The *Manager target* field shall indicate the IEEE address of the Data Management device transmitting this frame.

12581 **9.7.2.3.2.4 Manager endpoint**

12582 The *Manager endpoint* field shall indicate the source endpoint from which the Data Management device is transmitting  
12583 this frame.

12584 **9.7.2.3.2.5 When generated**

12585 This command is generated when a Data Management device wishes to connect to an 11073 agent device.

12586 This may be in response to receiving a connect status notification command from that agent device with the connect  
12587 status field set to RECONNECT\_REQUEST.

12588 **9.7.2.3.2.6 Effect on Receipt**

12589 On receipt of this command, a device shall first check if it is already connected by examining its *Connected*  
12590 attribute.

12591 If the tunnel is already connected then the device shall generate a connect status notification command  
12592 with status set to ALREADY\_CONNECTED and transmit it to the sender of this connect request frame. No  
12593 other attributes shall be affected, and no further processing shall be carried out.

12594 If the tunnel is not currently connected then the device shall copy the preemptible bit of connect control field  
12595 into the preemptible attribute, the idle timeout value into the idle timeout attribute, the manager target value  
12596 into the *ManagerTarget* attribute and the manager endpoint value into the *ManagerEndpoint* attribute.

12597 It shall set the connected attribute to TRUE, and generate a connect status notification command with status  
12598 set to CONNECTED and transmit it to the sender of this connect request frame.



12599 Finally, if the idle timeout field is set to a value other than 0xffff, the device shall set a timer for the timeout time  
 12600 indicated. This timer shall be restarted at any time that data is transmitted or received over the tunnel. If the timer  
 12601 expires then the device shall set the *Connected* attribute to FALSE and a connect status notification command with  
 12602 status DISCONNECTED shall be sent to the currently connected Data Management device. In order to continue to  
 12603 use the tunnel, the agent device shall send the Data Management device a further connect status notification command  
 12604 with status RECONNECT\_REQUEST, and wait for the Data Management device to respond.

12605 **9.7.2.3.3 Disconnect Request Command**

12606 The Disconnect Request command payload shall be formatted as illustrated in Figure 9-22.

<b>Octets</b>	8
<b>Data Type</b>	IEEE address
<b>Field Name</b>	Manager IEEE address

12607 **Figure 9-22 – Disconnect Request command payload**

12608 **9.7.2.3.3.1 Manager IEEE address**

12609 The *Manager IEEE address* field shall indicate the IEEE address of the Data Management device transmitting this  
 12610 frame.

12611 **9.7.2.3.3.2 When generated**

12612 This command is generated when a Data Management device wishes to disconnect a tunnel connection existing on an  
 12613 agent device.

12614 **9.7.2.3.3.3 Effect on Receipt**

12615 On receipt of this command, a device shall first check if it is already connected by examining its *Connected*  
 12616 attribute.

12617 If it is not currently connected then the device shall generate a connect status notification command with  
 12618 status set to DISCONNECTED and transmit it to the sender of this disconnect request frame. No other  
 12619 attributes shall be affected, and no further processing shall be carried out.

12620 If it is currently connected then the device shall check whether the requesting device is authorized to remove this  
 12621 connection. A device is authorized to remove the connection if the value of the manager IEEE address field is the  
 12622 same as the value in the *ManagerTarget* attribute or if the *Preemptible* attribute is set to TRUE.

12623 If the requester is not authorized then the device shall generate a connect status notification command with  
 12624 status set to NOT\_AUTHORIZED and transmit it to the sender of this disconnect request frame. No other  
 12625 attributes shall be affected, and no further processing shall be carried out.

12626 If the requester is authorized then the device shall initiate disconnection. A short period of time is permitted in order  
 12627 to allow the higher layer to finalize its activities, but within 12 seconds the device shall generate a connect status  
 12628 notification command with status set to DISCONNECTED and transmit it to the target indicated in the *ManagerTarget*  
 12629 attribute. The *Connected* attribute shall be set to FALSE and the tunnel shall be disconnected. The device shall now  
 12630 generate a further connect status notification command with status set to DISCONNECTED and transmit it to the  
 12631 sender of this disconnect request frame.

12632 **9.7.2.3.4 Connect Status Notification Command**

12633 The Connect Status Notification command payload shall be formatted as illustrated in Figure 9-23.

<b>Octets</b>	1
<b>Data Type</b>	enum8
<b>Field Name</b>	Connect status

Figure 9-23 – Connect Status Notification command payload

12634

12635 **9.7.2.3.4.1 Connect Status**12636 The *connect status* field shall be set to one of the values in Table 9-35:

12637

Table 9-35 – Connect status values

Value	Designation	Description
0x00	DISCONNECTED	Indicates that this agent device has been disconnected from the tunnel.
0x01	CONNECTED	Indicates that this agent device has been connected to the tunnel.
0x02	NOT_AUTHORIZED	Indicates that a request to disconnect the tunnel is not authorized from this requester at this time.
0x03	RECONNECT_REQUEST	Indicates that the agent device wishes the Data Management device to reconnect the tunnel.
0x04	ALREADY_CONNECTED	Indicates that the request to connect this tunnel has failed as the agent device is already connected.

12638 **9.7.2.3.4.2 When generated**12639 This command is generated by an agent device in response to a connect request command, disconnect command, or  
12640 in response to some other event that causes the tunnel to become connected or disconnected.

12641 It is also sent by the agent device to request the Data Management device to reconnect a tunnel.

12642 **9.7.2.3.4.3 Effect on Receipt**12643 On receipt of this command, a device shall be informed of the new status of the tunnel connection or of its attempt to  
12644 modify the status of the connection.12645 If the connect status field takes the value RECONNECT\_REQUEST then, depending on available resources being  
12646 available, the Data Management device should attempt to reconnect the tunnel by generating a connect request  
12647 command and transmitting it to the agent device sending this connect status notification command.12648 **9.7.2.4 Commands Generated**

12649 No cluster specific commands are generated by the server cluster.

12650 **9.7.3 Client**

12651 **9.7.3.1 Dependencies**

12652 Any endpoint that supports the 11073 Protocol Tunnel client cluster shall also support the Generic Tunnel client cluster  
12653 (see 9.2).

12654 **9.7.3.2 Attributes**

12655 The client cluster has no attributes.

12656 **9.7.3.3 Commands Received**

12657 The client does not receive any cluster specific commands.

12658 **9.7.3.4 Commands Generated**

12659 The cluster specific commands generated by the client cluster are listed in 9.7.2.3.

12660 In order to reduce the burden on implementations, some commands and attributes are conditionally mandated, as  
12661 follows:

- 12662 • Transmission of the transfer APDU command is mandatory.
- 12663 • Transmission of the connect request and disconnect request commands is optional unless specified  
12664 otherwise.
- 12665 • If the 11073 Protocol Tunnel server cluster implemented on the same endpoint implements any of the  
12666 *ManagerTarget*, *ManagerEndpoint*, *Connected*, *Preemptible* and *IdleTimeout* attributes, or implements  
12667 reception of the connect request or disconnect request commands, then transmission of the connect status  
12668 notification command is mandatory.
- 12669 • Transmission of non-cluster specific commands to manipulate attributes is optional unless specified  
12670 otherwise.



# CHAPTER 10 SMART ENERGY

12671

12672  
 12673  
 12674  
 12675

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

12676

## 10.1 General Description

12677

### 10.1.1 Introduction

12678  
 12679

The clusters specified in this chapter are for use typically in Smart Energy applications with associated security controls at the application layer. These clusters may be used in any application domain.

12680

### 10.1.2 Cluster List

12681  
 12682

This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of clarification. The clusters specified in this chapter are listed in Table 10-1.

12683

**Table 10-1. Smart Energy Clusters**

Cluster ID	Cluster Name	Description
0x0700	Price	Commands and attributes for reporting price
0x0701	Demand Response and Load Control	Commands and attributes for providing demand response and load control of devices
0x0702	Metering	Commands and attributes for reporting metering data
0x0703	Messaging	Commands and attributes for sending messages to devices
0x0704	Tunneling	Commands and attributes for establishing and using a tunnel between two devices
0x0800	Key Establishment	Commands and attributes for application level security establishment

12684

## 10.2 Price

12685

### 10.2.1 Overview

12686  
 12687

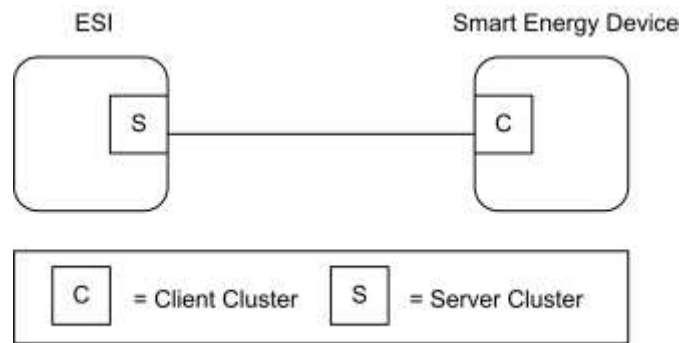
Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.

12688  
 12689  
 12690  
 12691  
 12692  
 12693

The Price Cluster provides the mechanism for communicating Gas, Energy, or Water pricing information within the premises. This pricing information is distributed to the ESI from either the utilities or from regional energy providers. The ESI conveys the information (via the Price Cluster mechanisms) to both Smart Energy devices in secure method and/or optionally conveys it anonymously in an unsecure to very simple devices that may not be part of the Smart Energy network. The mechanism for sending anonymous information is called the Anonymous Inter-PAN transmission mechanism.

12694

**Figure 10-1. Price Cluster Client Server Example**



12695

*Note: Device names are examples for illustration purposes only*

12696  
12697

Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream price management systems and subsequent data stores.

### 12698 10.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 12699 10.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEPR	Type 1 (client to server)

### 12700 10.2.1.3 Cluster Identifiers

Identifier	Name
0x0700	Price

## 12701 10.2.2 Server

### 12702 10.2.2.1 Dependencies

- 12703 • Events carried using this cluster include a timestamp with the assumption that target devices maintain a real  
12704 time clock. Devices can acquire and synchronize their internal clocks via the ZCL Time server.
- 12705 • If a device does not support a real time clock it is assumed that the device will interpret and utilize the  
12706 “Start Now” value within the Time field.
- 12707 • Anonymous Inter-PAN transmission mechanism.

### 12708 10.2.2.2 Attributes

12709 For convenience, the attributes defined in this cluster are arranged into sets of related attributes; each set can  
12710 contain up to 256 attributes. Attribute identifiers are encoded such that the most significant octet specifies the

12711 attribute set and the least significant octet specifies the attribute within the set. The currently defined attribute sets  
 12712 are listed in Table 10-2.

12713 **Note:** Price Cluster Attribute Set 0x03 in this revision of this specification is provisional and not certifiable. This  
 12714 feature set may change before reaching certifiable status in a future revision of this specification.

12715 **Table 10-2. Price Cluster Attribute Sets**

Attribute Set Identifier	Description
0x00	Tier Label
0x01	Block Threshold
0x02	Block Period
0x03	Commodity
0x04	Block Price Information
0x07	Billing Information Set

12716 **10.2.2.2.1 Tier Label Set**

12717 **Note:** Tier Label Set 0x06-0x0E in this revision of this specification are provisional and not certifiable. This  
 12718 feature set may change before reaching certifiable status in a future revision of this specification.

12719 **Table 10-3. Tier Label Attribute Set**

Id	Name	Type	Length	Access	Default	M/O
0x0000	<i>Tier1PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 1”	O
0x0001	<i>Tier2PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 2”	O
0x0002	<i>Tier3PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 3”	O
0x0003	<i>Tier4PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 4”	O
0x0004	<i>Tier5PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 5”	O
0x0005	<i>Tier6PriceLabel</i>	octstr	1 to 13 Octets	RW	“Tier 6”	O
0x0006	<i>Tier7PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 7”	O
0x0007	<i>Tier8PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 8”	O
0x0008	<i>Tier9PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 9”	O
0x0009	<i>Tier10PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 10”	O
0x000A	<i>Tier11PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 11”	O
0x000B	<i>Tier12PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 12”	O
0x000C	<i>Tier13PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 13”	O
0x000D	<i>Tier14PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 14”	O
0x000E	<i>Tier15PriceLabel</i>	octstr	1 to 13 Octets	R	“Tier 15”	O

12720 **10.2.2.2.1.1 TierNPriceLabel Attributes**

12721 The TierNPriceLabel attributes provide a method for utilities to assign a label to the Price Tier declared within the  
 12722 Publish Price command. The TierNPriceLabel attributes are a ZCL Octet String field capable of storing a 12  
 12723 character string (the first octet indicates length) encoded in the UTF-8 format. Example Tier Price Labels are  
 12724 “Normal”, “Shoulder”, “Peak”, “Real Time,” and “Critical”.

12725 **10.2.2.2.2 Block Threshold Set**

12726 The set of attributes shown in Table 10-4 provides remote access to the Price server Block Thresholds. Block  
 12727 Threshold values are crossed when the CurrentBlockPeriodConsumptionDelivered attribute value is greater than a  
 12728 BlockNThreshold attribute. The number of block thresholds is indicated by the Number of Block Thresholds field  
 12729 in the associated Publish Price command. The number of blocks is one greater than the number of thresholds.

12730 **Table 10-4. Block Threshold Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Access</b>	<b>Def</b>	<b>M/O</b>
0x0000	<i>Block1Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0001	<i>Block2Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0002	<i>Block3Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0003	<i>Block4Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0004	<i>Block5Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0005	<i>Block6Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0006	<i>Block7Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0007	<i>Block8Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0008	<i>Block9Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0009	<i>Block10Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x000A	<i>Block11Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x000B	<i>Block12Threshold</i>	uint48	0x000000000000 to xFFFFFFFFFFFFFFF	R	-	O
0x000C	<i>Block13Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x000D	<i>Block14Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x000E	<i>Block15Threshold</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O



12731 **10.2.2.2.1 BlockNThreshold**

12732 Attributes Block1Threshold through Block15Threshold represent the block threshold values for a given period  
 12733 (typically the billing cycle). These values may be updated by the utility on a seasonal or annual basis. The  
 12734 thresholds are established such that crossing the threshold of energy consumption for the present block activates the  
 12735 next higher block, which can affect the energy rate in a positive or negative manner. The values are absolute and  
 12736 always increasing. The values represent the threshold at the end of a block. The Unit of Measure will be based on the  
 12737 fields defined in the Publish Price command, the formatting being defined by attributes within the Block Period  
 12738 attribute set.

12739 **10.2.2.2.3 Block Period Set**

12740 The set of attributes shown in Table 10-5 provides remote access to the Price server Block Threshold period  
 12741 (typically the billing cycle) information.

12742 **Table 10-5. Block Period Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>StartofBlockPeriod</i>	UTC	-	R	-	O
0x0001	<i>BlockPeriodDuration</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0002	<i>ThresholdMultiplier</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0003	<i>ThresholdDivisor</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O

12743 **10.2.2.2.3.1 StartofBlockPeriod Attribute**

12744 The StartofBlockPeriod attribute represents the start time of the current block tariff period. A change indicates  
 12745 that a new Block Period is in effect (see sub-clause 10.2.4.3 for further details).

12746 **10.2.2.2.3.2 BlockPeriodDuration Attribute**

12747 The BlockPeriodDuration attribute represents the current block tariff period duration in minutes. A change indicates  
 12748 that only the duration of the current Block Period has been modified. A client device shall expect a new Block  
 12749 Period following the expiration of the new duration.

12750 **10.2.2.2.3.3 ThresholdMultiplier Attribute**

12751 ThresholdMultiplier provides a value to be multiplied against Threshold attributes. If present, this attribute must be  
 12752 applied to all Block Threshold values to derive values that can be compared against the  
 12753 CurrentBlockPeriodConsumptionDelivered attribute within the Metering cluster (see 10.4.2.2.1.13). This attribute  
 12754 must be used in conjunction with the ThresholdDivisor attribute. An attribute value of zero shall result in a  
 12755 unitary multiplier (0x000001).

12756 **10.2.2.2.3.4 ThresholdDivisor Attribute**

12757 ThresholdDivisor provides a value to divide the result of applying the ThresholdMultiplier attribute to Block  
 12758 Threshold values to derive values that can be compared against the CurrentBlockPeriodConsumptionDelivered  
 12759 attribute within the Metering cluster (see 10.4.2.2.1.13). This attribute must be used in conjunction with the  
 12760 ThresholdMultiplier attribute. An attribute value of zero shall result in a unitary divisor (0x000001).

12761 **10.2.2.2.4 Commodity Set**

12762 The set of attributes shown in Table 10-6 represents items that are associated with a particular commodity.

12763 **Note:** With the exception of the Standing Charge attribute, the Commodity Attribute Set in this revision of this  
 12764 specification is provisional and not certifiable. This feature set may change before reaching certifiable status in  
 12765 a future revision of this specification.

12766

**Table 10-6. Commodity Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Default</b>	<b>M/O</b>
0x0000	<i>CommodityType</i>	enum8	0x00 to 0xFF	R	-	O
0x0001	<i>StandingCharge</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0002	<i>ConversionFactor</i>	uint32	0x00000000 to 0xFFFFFFFF	R	0x10000000	O
0x0003	<i>ConversionFactorTrailingDigit</i>	map8		R	0x70	O
0x0004	<i>CalorificValue</i>	uint32	0x00000000 to 0xFFFFFFFF	R	0x2625A00	O
0x0005	<i>CalorificValueUnit</i>	enum8		R	0x1	O
0x0006	<i>CalorificValueTrailingDigit</i>	map8		R	0x60	O

**12767 10.2.2.2.4.1 CommodityType Attribute**

12768 CommodityType provides a label for identifying the type of pricing server present. The attribute is an enumerated  
 12769 value representing the commodity. The defined values are represented by the non-mirrored values (0-127) in the  
 12770 MeteringDeviceType attribute enumerations (refer to Table 10-42).

**12771 10.2.2.2.4.2 Standing Charge Attribute**

12772 The value of the Standing Charge is a daily fixed charge associated with supplying the commodity, measured in  
 12773 base unit of Currency with the decimal point located as indicated by the Trailing Digits field of a Publish Price  
 12774 command (see sub-clause 10.2.2.4.1). A value of 0xFFFFFFFF indicates field not used.

**12775 10.2.2.2.4.3 ConversionFactor Attribute**

12776 The conversion factor is used for gas meter and takes into account changes in the volume of gas based on  
 12777 temperature and pressure. The ConversionFactor attribute represents the current active value. The ConversionFactor  
 12778 is dimensionless. The default value for the ConversionFactor is 1, which means no conversion is applied. A price  
 12779 server can advertise a new/different value at any time.

**12780 10.2.2.2.4.4 ConversionFactorTrailingDigit Attribute**

12781 An 8-bit bitmap used to determine where the decimal point is located in the ConversionFactor attribute. The most  
 12782 significant nibble indicates the number of digits to the right of the decimal point. The least significant nibble is  
 12783 reserved. The ConversionFactorTrailingDigit attribute represents the current active value.

**12784 10.2.2.2.4.5 CalorificValue Attribute**

12785 The amount of heat generated when a given mass of fuel is completely burned. The CalorificValue is used to  
 12786 convert the measured volume or mass of gas into kWh. The CalorificValue attribute represents the current active  
 12787 value.

**12788 10.2.2.2.4.6 CalorificValueUnit Attribute**

12789 This attribute defines the unit for the CalorificValue. This attribute is an 8-bit enumerated field. The values and  
 12790 descriptions for this attribute are listed in Table 10-7 below. The CalorificValueUnit attribute represents the  
 12791 current active value.

12792

**Table 10-7. Values and Descriptions for the *CalorificValueUnit* Attribute**

Values	Description
0x01	MJ/m <sup>3</sup>
0x02	MJ/kg

12793

**10.2.2.2.4.7 CalorificValueTrailingDigit Attribute**

12794

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.

12795

12796

12797

**10.2.2.2.5 Block Price Information Set**

12798

The set of attributes shown in Table 10-8 provide remote access to the block prices. The Block Price

12799

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.

12800

12801

**Table 10-8. Block Price Information Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0000	<i>NoTierBlock1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0001	<i>NoTierBlock2Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0002	<i>NoTierBlock3Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x000N	<i>NoTierBlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x000F	<i>NoTierBlock16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0010	<i>Tier1Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0011	<i>Tier1Block2Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0012	<i>Tier1Block3Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x001N	<i>Tier1BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x001F	<i>Tier1Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0020	<i>Tier2Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x002N	<i>Tier2BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x002F	<i>Tier2Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0030	<i>Tier3Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x003N	<i>Tier3BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x003F	<i>Tier3Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0040	<i>Tier4Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x004N	<i>Tier4BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x004F	<i>Tier4Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x0050	<i>Tier5Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x5N	<i>Tier5BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

0x5F	<i>Tier5Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x60	<i>Tier6Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x6N	<i>Tier6BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x6F	<i>Tier6Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x70	<i>Tier7Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x7N	<i>Tier7BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x7F	<i>Tier7Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x80	<i>Tier8Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x8N	<i>Tier8BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x8F	<i>Tier8Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x90	<i>Tier9Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x9N	<i>Tier9BlockN+1Price ...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0x9F	<i>Tier9Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xA0	<i>Tier10Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xAN	<i>Tier10BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xAF	<i>Tier10Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xB0	<i>Tier11Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xBN	<i>Tier11BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xBF	<i>Tier11Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xC0	<i>Tier12Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xCN	<i>Tier12BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

0xCF	<i>Tier12Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xD0	<i>Tier13Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xDN	<i>Tier13BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xDF	<i>Tier13Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xE0	<i>Tier14Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xEN	<i>Tier14BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xEF	<i>Tier14Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xF0	<i>Tier15Block1Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xFN	<i>Tier15BlockN+1Price...</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O
0xFF	<i>Tier15Block16Price</i>	uint32	0x00000000 to 0xFFFFFFFF	R	-	O

12802 **10.2.2.2.5.1 TierNBlockNPrice Attributes**

12803 Attributes PriceNoTierBlock1 through PriceTier15Block16 represent the price of Energy, Gas, or Water delivered  
 12804 to the premises (i.e. delivered to the customer from the utility) at a specific price tier as defined by a TOU  
 12805 schedule, Block Threshold or a real time pricing period. If optionally provided, attributes shall be initialized prior  
 12806 to the issuance of associated Publish Price commands (see sub-clause 10.2.2.4.1). The expected practical limit for  
 12807 the number of PriceTierNBlockN attributes supported is 32. The Unit of Measure, Currency and Trailing Digits that  
 12808 apply to this attribute should be obtained from the appropriate fields in a Publish Price command.

12809 **10.2.2.2.6 Billing Information Attribute Set**

12810 The set of attributes shown in Table 10-9 provides remote access to the Price server Billing information.

12811 **Table 10-9. Billing Information Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0000	<i>CurrentBillingPeriodStart</i>	UTC	0x00000000 to 0xFFFFFFFF	R	-	O
0x0001	<i>CurrentBillingPeriodDuration</i>	uint24	0x000000 to 0xFFFFF	R	-	O

12812 **10.2.2.2.6.1 CurrentBillingPeriodStart Attribute**

12813 The CurrentBillingPeriodStart attribute represents the start time of the current billing period.

12814 **10.2.2.2.6.2 CurrentBillingPeriodDuration Attribute**

12815 The CurrentBillingPeriodDuration attribute represents the current billing period duration in minutes.

### 12816 10.2.2.3 Commands Received

12817 The server side of the Price cluster is capable of receiving the commands listed in Table 10-10.

12818 **Table 10-10. Received Command IDs for the Price Cluster**

Command Identifier Field Value	Description	M/O
0x00	<i>Get Current Price</i>	M
0x01	<i>Get Scheduled Prices</i>	O
0x02	<i>Price Acknowledgement</i>	M for 1.1 and later devices
0x03	<i>Get Block Period(s)</i>	O
0x04	<i>GetConversionFactor</i>	O
0x05	<i>GetCalorificValue</i>	O

#### 12819 10.2.2.3.1 Get Current Price Command

12820 This command initiates a Publish Price command (see sub-clause 10.2.2.4.1) for the current time.

##### 12821 10.2.2.3.1.1 Payload Format

12822 The payload of the Get Current Price command is formatted as shown in Figure 10-2.

12823 **Figure 10-2. The Format of the Get Current Price Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Command Options

##### 12824 10.2.2.3.1.1.1 Payload Details

12825 **The Command Options Field:** The command options field is 8 Bits in length and is formatted as a bit field as shown  
 12826 in Figure 10-3.

12827 **Figure 10-3. Get Current Price Command Options Field**

<b>Bits</b>	0	1 to 7
<b>Field Name</b>	Requestor Rx On When Idle	Reserved

12828  
 12829 **The Requestor Rx On When Idle Sub-field:** The Requestor Rx On When Idle sub-field has a value of 1 if the  
 12830 requestor’s receiver may be, for all practical purposes, enabled when the device is not actively transmitting,  
 12831 thereby making it very likely that regular broadcasts of pricing information will be received by this device, and 0  
 12832 otherwise.

12833 A device that publishes price information may use the value of this bit, as received from requestors in its  
 12834 neighborhood, to determine publishing policy. For example, if a device makes a request for current pricing  
 12835 information and the requestor Rx on when idle sub-field of the GetCurrentPrice command payload has a value of 1

12836 (indicating that the device will be likely to receive regular price messages), then the receiving device may store  
12837 information about the requestor and use it in future publishing operations.

#### 12838 **10.2.2.3.1.2 Effect on Receipt**

12839 On receipt of this command, the device shall send a Publish Price command (sub-clause 10.2.2.4.1) for the currently  
12840 scheduled time. Please note: The PublishPrice command is sent out on the network from which the  
12841 GetCurrentPrice command was received (either the Inter-Pan or SE network). Example: If the GetCurrentPrice  
12842 command is received on the Inter-Pan network, the ESI shall respond on the Inter-Pan. If the GetCurrentPrice  
12843 command is received on the SE Network, the ESI shall respond to the device requesting the pricing information.

#### 12844 **10.2.2.3.2 Get Scheduled Prices Command**

12845 This command initiates a Publish Price command (see sub-clause 10.2.2.4.1) for available price events. A server  
12846 device shall be capable of storing five price events at a minimum.

##### 12847 **10.2.2.3.2.1 Payload Details**

12848 The Get Scheduled Prices command payload shall be formatted as illustrated in Figure 10-4.

12849 **Figure 10-4. Format of the *Get Scheduled Prices* Command Payload**

<b>Octets</b>	4	1
<b>Data Type</b>	UTC	uint8
<b>Field Name</b>	Start Time (M)	Number of Events (M)

12850  
12851 **Start Time (mandatory):** UTC Timestamp representing the minimum ending time for any scheduled or  
12852 currently active pricing events to be resent. If a command has a Start Time of 0x00000000, replace that Start Time  
12853 with the current time stamp.

12854 **Number of Events (mandatory):** Represents the maximum number of events to be sent. A value of 0 would  
12855 indicate all available events are to be returned. Example: Number of Events = 1 would return the first event  
12856 with an EndTime greater than or equal to the value of Start Time field in the Get Scheduled Prices command.  
12857 (EndTime would be StartTime plus Duration of the event listed in the device's event table).

##### 12858 **10.2.2.3.2.2 When Generated**

12859 This command is generated when the client device wishes to verify the available Price Events or after a loss of  
12860 power/reset occurs and the client device needs to recover currently active, scheduled, or expired Price Events.

12861 A ZCL Default Response with status NOT\_FOUND shall be returned if there are no events available.

##### 12862 **10.2.2.3.2.3 Effect on Receipt**

12863 On receipt of this command, the device shall send a Publish Price command (see sub-clause 10.2.2.4.1) for all  
12864 currently scheduled price events. Please note: The Publish Price command is sent out on the network from which  
12865 the GetScheduledPrices command was received (either the Inter-Pan or SE network). Example: If the  
12866 GetScheduledPrices command is received on the Inter-Pan network, the ESI shall respond on the Inter-Pan. If the  
12867 GetScheduledPrices command is received on the SE Network, the ESI shall respond to the device requesting the  
12868 pricing information.

#### 12869 **10.2.2.3.3 Price Acknowledgement Command**

12870 The Price Acknowledgement command described in Figure 10-5 provides the ability to acknowledge a  
12871 previously sent Publish Price command. It is mandatory for 1.1 and later devices. For SE 1.0 devices, the command  
12872 is optional.



12873 **10.2.2.3.3.1 Payload Format**

12874 **Figure 10-5. Format of the Price Acknowledgement Command Payload**

<b>Octets</b>	4	4	4	1
<b>Data Type</b>	uint32	uint32	UTC	map8
<b>Field Name</b>	Provider ID (M)	Issuer Event ID (M)	Price Ack Time (M)	Control (M)

12875 **10.2.2.3.3.1.1 Payload Details**

12876 **Provider ID (mandatory):** An unsigned 32-bit field containing a unique identifier for the commodity provider.

12877 **Issuer Event ID (mandatory):** Unique identifier generated by the commodity provider.

12878 **Price Ack Time (mandatory):** Time price acknowledgement generated.

12879 **Control (mandatory):** Identifies the Price Control or Block Period Control options for the event. The values for  
 12880 this field are described in Figure 10-9 and Figure 10-10.

12881 **10.2.2.3.3.2 When Generated**

12882 This command is generated on receipt of a Publish Price command when the Price Control field of that Publish  
 12883 Price command indicates that a Price Acknowledgement is required (see sub-clause 10.2.2.4.1 for further details).

12884 **10.2.2.3.4 Get Block Period(s) Command**

12885 This command initiates a Publish Block Period command (see sub-clause 10.2.2.4.2) for the currently scheduled  
 12886 block periods. A server device shall be capable of storing at least two commands, the current period and a period to  
 12887 be activated in the near future.

12888 **Note:** The Get Block Period(s) command in this revision of this specification is provisional and not certifiable.  
 12889 This feature set may change before reaching certifiable status in a future revision of this specification.

12890 **10.2.2.3.4.1 Payload Format**

12891 **Figure 10-6. Format of the Get Block Period(s) Command Payload**

<b>Octets</b>	4	1
<b>Data Type</b>	UTC	uint8
<b>Field Name</b>	Start Time (M)	Number of Events (M)

12892 **10.2.2.3.4.1.1 Payload Details**

12893 **Start Time (mandatory):** UTC Timestamp representing the minimum ending time for any scheduled or  
 12894 currently block period events to be resent. If a command has a Start Time of 0x00000000, replace that Start  
 12895 Time with the current time stamp.

12896 **Number of Events (mandatory):** An 8 bit integer which indicates the maximum number of Publish Block Period  
 12897 commands that can be sent. Example: Number of Events = 1 would return the first event with an EndTime greater  
 12898 than or equal to the value of Start Time field in the GetBlockPeriod(s) command. (EndTime would be StartTime plus  
 12899 Duration of the event listed in the device’s event table). Number of Events = 0 would return all available Publish  
 12900 Block Periods, starting with the current block in progress.

12901 **10.2.2.3.4.2 When Generated**

12902 This command is generated when the client device wishes to verify the available Block Period events or after a  
 12903 loss of power/reset occurs and the client device needs to recover currently active or scheduled Block Periods.

12904 A ZCL Default response with status NOT\_FOUND shall be returned if there are no events available.

#### 12905 **10.2.2.3.4.3 Effect on Receipt**

12906 On receipt of this command, the device shall send a Publish Block Period command (sub-clause 10.2.2.4.2) for all  
12907 currently scheduled periods, up to the maximum number of commands specified.

### 12908 **10.2.2.3.5 GetConversionFactor Command**

12909 This command initiates a PublishConversionFactor command for the scheduled conversion factor updates. A server  
12910 device shall be capable of storing at least two instances, the current and next instance to be activated in the near  
12911 future (if available).

12912 **Note:** The GetConversionFactor command in this revision of this specification is provisional and not certifiable.  
12913 This feature set may change before reaching certifiable status in a future revision of this specification.

#### 12914 **10.2.2.3.5.1 Payload Format**

12915 **Figure 10-7. Format of the *GetConversionFactor* Command Payload**

<b>Octets</b>	4	4
<b>Data Type</b>	UTC	uint8
<b>Field Name</b>	Start Time	Number of Events

#### 12916 **10.2.2.3.5.2 Payload Details**

12917 **Start Time (mandatory):** UTC Timestamp to select active and scheduled events to be returned by the  
12918 corresponding PublishConversionFactor command. If command has a Start Time of 0x00000000, replace that  
12919 Start Time with the current time stamp.

12920 **Number of Events (mandatory):** An 8-bit integer which represents the maximum number of  
12921 PublishConversionFactor commands to be sent. A value of 0 would indicate all available PublishConversionFactor  
12922 commands shall be returned. The first returned PublishConversionFactor command shall be the instance which is  
12923 active or becomes active at the stated Start Time. If more than one instance is requested, the active and scheduled  
12924 instances shall be sent with ascending ordered StartTime.

### 12925 **10.2.2.3.6 GetCalorificValue Command**

12926 This command initiates a PublishCalorificValue command for the scheduled calorific value updates. A server  
12927 device shall be capable of storing at least two instances, the current and next instance to be activated in the  
12928 near future (if available).

12929 **Note:** The GetCalorificValue command in this revision of this specification is provisional and not certifiable. This  
12930 feature set may change before reaching certifiable status in a future revision of this specification.

#### 12931 **10.2.2.3.6.1 Payload Format**

12932

**Figure 10-8. Format of the *GetCalorificValue* Command Payload**

<b>Octets</b>	4	1
<b>Data Type</b>	UTC	uint8
<b>Field Name</b>	Start Time	Number of Events

12933 **10.2.2.3.6.2 Payload Details**

12934 **Start Time (mandatory):** UTC Timestamp to select active and scheduled events to be returned by the  
 12935 corresponding PublishCalorificValue command. If the command has a Start Time of 0x00000000, replace that  
 12936 Start Time with the current time stamp.

12937 **Number of Events (mandatory):** An 8-bit integer which represents the maximum number of  
 12938 PublishCalorificValue commands to be sent. A value of 0 would indicate all available PublishCalorificValue  
 12939 commands shall be returned. The first returned PublishCalorificValue command shall be the instance which is  
 12940 active at the stated Start Time. If more than one instance is requested, the active and scheduled instances shall be  
 12941 sent with ascending ordered Start Time.

12942 **10.2.2.4 Commands Generated**

12943 The server side of the Price cluster is capable of generating the commands listed in Table 10-11.

12944 **Table 10-11. Generated Command IDs for the Price Cluster**

<b>Command Identifier Field Value</b>	<b>Description</b>	<b>M/O</b>
0x00	<i>Publish Price</i>	M
0x01	<i>Publish Block Period</i>	O
0x02	<i>Publish Conversion Factor</i>	O
0x03	<i>Publish Calorific Value</i>	O

12945 **10.2.2.4.1 Publish Price Command**

12946 The Publish Price command is generated in response to receiving a Get Current Price command (see sub-clause  
 12947 10.2.2.3.1), in response to a Get Scheduled Prices command (see sub-clause 10.2.2.3.2), and when an update to the  
 12948 pricing information is available from the commodity provider, either before or when a TOU price becomes active .  
 12949 Additionally the Publish Price Command is generated as specified in sub-clause 10.2.4.3 when Block Pricing is in  
 12950 effect.

12951 When a Get Current Price or Get Scheduled Prices command is received over a Smart Energy network, the  
 12952 Publish Price command should be sent unicast to the requester. In the case of an update to the pricing  
 12953 information from the commodity provider, the Publish Price command should be unicast to all individually  
 12954 registered devices implementing the Price Cluster on the Smart Energy network. When responding to a request  
 12955 via the Inter- PAN SAP, the Publish Price command should be broadcast to the PAN of the requester after a  
 12956 random delay between 0 and 0.5 seconds, to avoid a potential broadcast storm of packets.

12957 Devices capable of receiving this command must be capable of storing and supporting at least two pricing  
 12958 information instances, the current active price and the next price. By supporting at least two pricing information  
 12959 instances, receiving devices will allow the Publish Price command generator to publish the next pricing information  
 12960 during the current pricing period.

12961 Nested and overlapping Publish Price commands are not allowed. The current active price will be replaced if new  
 12962 price information is received by the ESI. In the case of overlapping events, the event with the newer Issuer Event ID

12963 takes priority over all nested and overlapping events. All existing events that overlap, even partially, should be  
 12964 removed. The only exception to this is that if an event with a newer Issuer Event ID overlaps with the end of the  
 12965 current active price but is not yet active, the active price is not deleted but its duration is modified to 0xFFFF  
 12966 (until changed) so that the active price ends when the new event begins.

#### 12967 10.2.2.4.1.1 Payload Format

12968 The PublishPrice command payload shall be formatted as illustrated in Figure 10-9.

12969 **Figure 10-9. Format of the Publish Price Command Payload**

<b>Octets</b>	4	1-13	4	4	1	2	1
<b>Data Type</b>	uint32	octstr	uint32	UTC	enum8	uint16	map8
<b>Field Name</b>	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)

12970

<b>Octets</b>	1	4	2	4	1	4	1
<b>Data Type</b>	map8	UTC	uint16	uint32	uint8	uint32	uint8
<b>Field Name</b>	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)

12971

<b>Octets</b>	4	1	1	1 <sup>b</sup>	1 <sup>c</sup>
<b>Data Type</b>	uint32	enum8	map8	8 bit integer	map8
<b>Field Name</b>	Alternate Cost Delivered (O)	Alternate Cost Unit (O)	Alternate Cost Trailing Digit(O)	Number of Block Thresholds (O)	Price Control (O)

12972

12973 **Note:** M = Mandatory field, O = Optional field. **All fields must be present in the payload.** Optional fields will be  
 12974 marked with specific values to indicate they are not being used.

12975 **Provider ID (mandatory):** An unsigned 32-bit field containing a unique identifier for the commodity provider.  
 12976 This field is thought to be useful in deregulated markets where multiple commodity providers may be available.

12977 **Rate Label (mandatory):** A ZCL Octet String field capable of storing a 12 character string (the first octet  
 12978 indicates length) containing commodity provider- specific information regarding the current billing rate. The String  
 12979 shall be encoded in the UTF-8 format. This field is thought to be useful when a commodity provider may have  
 12980 multiple pricing plans.

12981 **Issuer Event ID (mandatory):** Unique identifier generated by the commodity provider. When new pricing  
 12982 information is provided that replaces older pricing information for the same time period, this field allows devices to  
 12983 determine which information is newer. It is expected that the value contained in this field is a unique number  
 12984 managed by upstream servers or a UTC based time stamp (UTC data type) identifying when the Publish Price  
 12985 command was issued. Thus, newer pricing information will have a value in the Issuer Event ID field that is larger  
 12986 than older pricing information.

12987 **Current Time (mandatory):** A UTC field containing the current time as determined by the device. This field is  
 12988 thought to be useful to provide an extra value-added feature for the broadcast price signals.

12989 **Unit of Measure (mandatory):** An 8-bit enumeration field identifying the commodity as well as its base unit of  
 12990 measure. The enumeration used for this field shall match one of the UnitOfMeasure values using a pure binary  
 12991 format as defined in the Metering cluster (see sub-clause 0).

12992 **Currency (mandatory):** An unsigned 16-bit field containing identifying information concerning the local unit of  
 12993 currency used in the price field. This field is thought to be useful for displaying the appropriate symbol for a currency  
 12994 (i.e.: \$).

12995 The value of the currency field should match the values defined by ISO 4217.

12996 **Price Trailing Digit and Price Tier (mandatory):** An 8-bit field used to determine where the decimal point is  
 12997 located in the price field and to indicate the current pricing tier as chosen by the commodity provider. The most  
 12998 significant nibble is the Trailing Digit sub-field which indicates the number of digits to the right of the decimal point.  
 12999 The least significant nibble is an enumerated field containing the current Price Tier. Valid values for the Price Tier sub-  
 13000 field are from 1 to 15 reflecting the least expensive tier (1) to the most expensive tier (15). A value of zero indicates  
 13001 no price tier is in use. This sub-field also references the associated TierPriceLabel attribute assigned to the Price Tier.  
 13002 Table 10-12 depicts the assignments.

13003 **Note:** Values for Price Tier listed above 0x06 in this revision of this specification are provisional and not  
 13004 certifiable. This number of fields may change before reaching certifiable status in a future revision of this  
 13005 specification.

13006 **Table 10-12. Price Tier Sub-field Enumerations**

Enumerated Value	Price Tier
0x0	No Tier Related
0x1	Reference <i>Tier1PriceLabel</i>
0x2	Reference <i>Tier2PriceLabel</i>
0x3	Reference <i>Tier3PriceLabel</i>
0x4	Reference <i>Tier4PriceLabel</i>
0x5	Reference <i>Tier5PriceLabel</i>
0x6	Reference <i>Tier6PriceLabel</i>
0x7	Reference <i>Tier7PriceLabel</i>
0x8	Reference <i>Tier8PriceLabel</i>
0x9	Reference <i>Tier9PriceLabel</i>
0xA	Reference <i>Tier10PriceLabel</i>
0xB	Reference <i>Tier11PriceLabel</i>
0xC	Reference <i>Tier12PriceLabel</i>
0xD	Reference <i>Tier13PriceLabel</i>
0xE	Reference <i>Tier14PriceLabel</i>
0xF	Reference <i>Tier15PriceLabel</i>

13007

13008 **Number of Price Tiers & Register Tier (mandatory):** An 8-bit bitmap where the most significant nibble is an  
 13009 enumerated sub-field representing the maximum number of price tiers available, and the least significant nibble is  
 13010 an enumerated sub-field indicating the register tier used with the current Price Tier. Valid values for the Number of  
 13011 Price Tiers sub-field are from 0 to 15 reflecting no tiers in use (0) to fifteen tiers available (15).

13012 The Register Tier values correlates which CurrentTierNSummationDelivered attribute, found in sub-clause  
 13013 10.4.2.2.2 is accumulating usage information. Both attributes can be used to calculate and display usage and  
 13014 subsequent costs. Register Tier enumerated values are listed in Table 10-13.

13015 **Note:** Values for Register Tier Sub-field Enumerations listed above 0x06 in this revision of this specification are  
 13016 provisional and not certifiable. This number of fields may change before reaching certifiable status in a future  
 13017 revision of this specification.

13018 **Table 10-13. Register Tier Sub-field Enumerations**

Enumerated Value	Register Tier
0x0	No Tier Related
0x1	Usage accumulating in <i>CurrentTier1SummationDelivered</i> attribute
0x2	Usage accumulating in <i>CurrentTier2SummationDelivered</i> attribute
0x3	Usage accumulating in <i>CurrentTier3SummationDelivered</i> attribute
0x4	Usage accumulating in <i>CurrentTier4SummationDelivered</i> attribute
0x5	Usage accumulating in <i>CurrentTier5SummationDelivered</i> attribute
0x6	Usage accumulating in <i>CurrentTier6SummationDelivered</i> attribute
0x7	Usage accumulating in <i>CurrentTier7SummationDelivered</i> attribute
0x8	Usage accumulating in <i>CurrentTier8SummationDelivered</i> attribute
0x9	Usage accumulating in <i>CurrentTier9SummationDelivered</i> attribute
0xA	Usage accumulating in <i>CurrentTier10SummationDelivered</i> attribute
0xB	Usage accumulating in <i>CurrentTier11SummationDelivered</i> attribute
0xC	Usage accumulating in <i>CurrentTier12SummationDelivered</i> attribute
0xD	Usage accumulating in <i>CurrentTier13SummationDelivered</i> attribute
0xE	Usage accumulating in <i>CurrentTier14SummationDelivered</i> attribute
0xF	Usage accumulating in <i>CurrentTier15SummationDelivered</i> attribute

13019  
 13020 **Start Time (mandatory):** A UTC field to denote the time at which the price signal becomes valid. A Start Time of  
 13021 0x00000000 is a special time denoting “now.”

13022 If the device would send a price with a Start Time of now, adjust the Duration In Minutes field to correspond to the  
 13023 remainder of the price.

13024 **Duration In Minutes (mandatory):** An unsigned 16-bit field used to denote the amount of time in minutes after  
 13025 the Start Time during which the price signal is valid. Maximum value means “until changed”. If Block Charging  
 13026 only is in use (see sub-clause 10.2.4.3 for further details), the Duration in Minutes field of the Publish Price  
 13027 command shall be set to 0xFFFF indicating the price is valid “until changed”.

- 13028 **Price (mandatory):** An unsigned 32-bit field containing the price of the commodity measured in base unit of  
 13029 Currency per Unit of Measure with the decimal point located as indicated by the Price Trailing Digit field  
 13030 when the commodity is delivered to the premises.
- 13031 **Price Ratio (optional):** An unsigned 8-bit field that gives the ratio of the price denoted in the Price field to the  
 13032 “normal” price chosen by the commodity provider. This field is thought to be useful in situations where client  
 13033 devices may simply be interested in pricing levels or ratios. The value in this field should be scaled by a factor of  
 13034 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  
 13035 value.
- 13036 **Generation Price (optional):** An unsigned 32-bit field containing the price of the commodity measured in base unit  
 13037 of Currency per Unit of Measure with the decimal point located as indicated by the Price Trailing Digit field when  
 13038 the commodity is received from the premises. An example use of this field is in energy markets where the price of  
 13039 electricity from the grid is different than the price of electricity placed on the grid. A value of 0xFFFFFFFF  
 13040 indicates the field is not used.
- 13041 **Generation Price Ratio (optional):** An unsigned 8-bit field that gives the ratio of the price denoted in the Generation  
 13042 Price field to the “normal” price chosen by the commodity provider. This field is thought to be useful in situations  
 13043 where client devices may simply be interested in pricing levels or ratios. The value in this field should be scaled by  
 13044 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  
 13045 is an invalid value.
- 13046 **Alternate Cost Delivered (optional):** An unsigned 32 Integer field that provides a mechanism to describe an  
 13047 alternative measure of the cost of the energy consumed. An example of an Alternate Cost might be the emissions of  
 13048 CO<sub>2</sub> for each kWh of electricity consumed providing a measure of the environmental cost. Another example is the  
 13049 emissions of CO<sub>2</sub> for each cubic meter of gas consumed (for gas metering). A different value for each price tier may  
 13050 be provided which can be used to reflect the different mix of generation that is associated with different TOU rates.  
 13051 A value of 0xFFFFFFFF indicates the field is not used.
- 13052 **Alternate Cost Unit (optional):** An 8-bit enumeration identifying the unit (as specified in Table 10-14) for the  
 13053 Alternate Cost Delivered field. A value of 0xFF indicates the field is not used.

Table 10-14. Alternate Cost Unit Enumerations

Values	Description
0x01	Kg of CO <sub>2</sub> per unit of measure

- 13054
- 13055
- 13056 **Alternate Cost Trailing Digit (optional):** An 8-bit bitmap field used to determine where the decimal point is  
 13057 located in the alternate cost field. The most significant nibble indicates the number of digits to the right of the  
 13058 decimal point. The least significant nibble is reserved. A value of 0xFF indicates the field is not used.
- 13059 **Number of Block Thresholds (optional):** An 8-bit integer which indicates the number of block thresholds  
 13060 available. Valid values are from 0 to 15 reflecting no blocks in use (0) to 15 block thresholds available (15). A  
 13061 value of 0xFF indicates field not used. Any value between 1 and 15 indicates that Block Pricing shall be used, see  
 13062 sub-clause 10.2.4.3 for further details.
- 13063 **Price Control (optional):** Identifies additional control options for the price event. A value of 0x00 indicates field  
 13064 not used. Note that for Smart Energy 1.1 and later devices, the Price Acknowledgement command is mandatory,  
 13065 but for SE 1.0 devices, it was optional, so the sender of the Publish Price command should not rely on receiving a  
 13066 Price Acknowledgment command even if the Price Acknowledgement bit in the Price Control Field is set.
- 13067 The BitMap for this field is described in Table 10-15.

13068

**Table 10-15. Price Control Field BitMap**

Bit	Description
0	1=Price Acknowledgement required 0=Price Acknowledgement not required

13069 **10.2.2.4.1.2 Effect on Receipt**

13070 On receipt of this command, the device is informed of a price event for the specific provider, commodity, and  
13071 currency indicated.

13072 Should the device choose to change behavior based on the price event, the change of behavior should occur after a  
13073 random delay between 0 and 5 minutes, to avoid potential spikes that could occur as a result of coordinated  
13074 behavior changes. Likewise, should a device choose to change behavior based on the expiration of the price event,  
13075 the change in behavior should occur after a random delay between 0 and 5 minutes.,

13076 **10.2.2.4.2 Publish Block Period Command**

13077 The Publish Block Period command is generated in response to receiving a Get Block Period(s) command (see  
13078 sub-clause 10.2.2.3.4) or when an update to the block tariff schedule is available from the commodity provider.  
13079 When the Get Block Period(s) command is received over the Smart Energy network, the Publish Block Period  
13080 command(s) should be sent unicast to the requestor. In the case of an update to the block tariff schedule from the  
13081 commodity provider, the Publish Block Period command should be unicast to all individually registered devices  
13082 implementing the Price Cluster on the Smart Energy network.

13083 Devices capable of receiving this command must be capable of storing and supporting two block periods, the  
13084 current active block and the next block. By supporting two block periods, receiving devices will allow the  
13085 Publish Block Period command generator to publish the next block information during the current block  
13086 period.

13087 **Note:** The Publish Block Period command in this revision of this specification is provisional and not certifiable.  
13088 This feature may change before reaching certifiable status in a future revision of this specification.

13089 **10.2.2.4.2.1 Payload Format**

13090 **Figure 10-10. Format of the Publish Block Period Command Payload**

<b>Octets</b>	4	4	4	3	1	1
<b>Data Type</b>	uint32	uint32	UTC	uint24	map8	map8
<b>Field Name</b>	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)

13091

13092 **Note:** M = Mandatory field, O = Optional field. **All fields shall be present in the payload.** Optional fields will be  
13093 marked with specific values to indicate they are not being used.

13094 **Provider ID (mandatory):** An unsigned 32-bit field containing a unique identifier for the commodity provider.  
13095 This field is thought to be useful in deregulated markets where multiple commodity providers may be available.

13096 **Issuer Event ID (mandatory):** Unique identifier generated by the commodity provider. When new block period  
13097 information is provided that replaces older information for the same period, this field allows devices to  
13098 determine which information is newer. It is expected that the value contained in this field is a unique number  
13099 managed by upstream servers or a UTC based time stamp (UTC data type) identifying when the Publish Block



13100 Period command was issued. Thus, newer block period information will have a value in the Issuer Event ID field that  
 13101 is larger than older block information.

13102 **Block Period Start Time (mandatory):** A UTC field to denote the time at which the block tariff period starts. A  
 13103 start time of 0x00000000 is a special time denoting “now”. If the device would send an event with a Start Time  
 13104 of now, adjust the Duration In Minutes field to correspond to the remainder of the event.

13105 **Block Period Duration In Minutes (mandatory):** An unsigned 24-bit field to denote the block tariff period in  
 13106 minutes. Maximum value (0xFFFFF) means 'until changed'.

13107 **Number of Price Tiers and Number of Block Thresholds (mandatory):** An 8-bit bitmap where the most  
 13108 significant nibble is an enumerated sub-field representing the maximum number of price tiers available, and the least  
 13109 significant nibble is an enumerated sub-field indicating the number of block thresholds available. Valid values  
 13110 for the Number of Price Tiers sub-field are from 0 to 15 reflecting no tiers in use (0) to fifteen tiers available (15).  
 13111 Valid values for the Number of Block Thresholds sub-field are from 0 to 15 reflecting no blocks in use (0) to 15  
 13112 block thresholds available (15).

13113 **Block Period Control (optional):** Identifies additional control options for the block period event. A value of  
 13114 0x00 indicates field not used.

13115 The BitMap for this field is described in Table 10-16.

13116 **Table 10-16. Block Period Control Field BitMap**

Bit	Description
0	1=Price Acknowledgement required 0=Price Acknowledgement not required
1	1=Repeating Block 0=Non Repeating Block

13117

13118 **Repeating Block:** Indicates whether a block period repeats on expiry.

13119 **10.2.2.4.3 PublishConversionFactor Command**

13120 The PublishConversionFactor command is sent in response to a GetConversionFactor command or if a new  
 13121 conversion factor is available.

13122 Clients shall be capable of storing at least two instances of the Calorific Value, the currently active one and the next  
 13123 one.

13124 **Note:** The PublishConversionFactor command in this revision of this specification is provisional and not  
 13125 certifiable. This feature may change before reaching certifiable status in a future revision of this specification.

13126 **10.2.2.4.3.1 Payload Format**

13127 **Figure 10-11. Format of the PublishConversionFactor Command Payload**

<b>Octets</b>	4	4	4	1
<b>Data Type</b>	uint32	UTC	uint32	map8
<b>Field Name</b>	Issuer Event ID (M)	Start Time (M)	Conversion Factor (M)	Conversion Factor Trailing Digit (M)

13128 **10.2.2.4.3.2 Payload Details**

13129 **Issuer Event ID (mandatory):** Unique identifier generated by the commodity provider.

13130 **Start Time (mandatory):** A UTC field to denote the time at which the value becomes valid. The value remains valid  
13131 until replaced by a newer one.

13132 **Conversion Factor (mandatory):** See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.3).

13133 **Conversion Factor Trailing Digit (mandatory):** See Price Cluster Commodity attributes (see sub-clause  
13134 10.2.2.2.4.4).

#### 13135 **10.2.2.4.4 PublishCalorificValue Command**

13136 The PublishCalorificValue command is sent in response to a GetCalorificValue command or if a new calorific  
13137 value is available. Clients shall be capable of storing at least two instances of the Calorific Value, the currently  
13138 active one and the next one.

13139 **Note:** The PublishCalorificValue command in this revision of this specification is provisional and not certifiable.  
13140 This feature may change before reaching certifiable status in a future revision of this specification.

##### 13141 **10.2.2.4.4.1 Payload Format**

13142 **Figure 10-12. Format of the *PublishCalorificValue* Command Payload**

<b>Octets</b>	4	4	4	1	1
<b>Data Type</b>	uint32	UTC	uint32	enum8	map8
<b>Field Name</b>	Issuer Event ID (M)	Start Time (M)	Calorific Value (M)	Calorific Value Unit (M)	Calorific Value Trailing Digit (M)

##### 13143 **10.2.2.4.4.2 Payload Details**

13144 **Issuer Event ID (mandatory):** Unique identifier generated by the commodity provider.

13145 **Start Time (mandatory):** A UTC field to denote the time at which the value becomes valid. The value remains valid  
13146 until replaced by a newer one.

13147 **Calorific Value (mandatory):** See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.5).

13148 **Calorific Value Unit (mandatory):** See Price Cluster Commodity attributes (see sub-clause 10.2.2.2.4.6).

13149 **Calorific Value Trailing Digit (mandatory):** See Price Cluster Commodity attributes (see sub-clause  
13150 10.2.2.2.4.7).

## 13151 **10.2.3 Client**

### 13152 **10.2.3.1 Dependencies**

13153 Events carried using this cluster include a timestamp with the assumption that target devices maintain a real  
13154 time clock. Devices can acquire and synchronize their internal clocks via the ZCL Time server.

13155 If a device does not support a real time clock it is assumed that the device will interpret and utilize the “Start  
13156 Now” 0x00000000 value within the Time field.

13157 Anonymous Inter-PAN transmission mechanism.

13158 **Note:** The Price Client Cluster Attributes in this revision of this specification are provisional and not certifiable.  
13159 These features may change before reaching certifiable status in a future revision of this specification.

13160 **10.2.3.2 Attributes**

13161 **Table 10-17. Price Client Cluster Attributes**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>PriceIncreaseRandomizeMinutes</i>	uint8	0x00 to 0x3C	RW	0x05	O
0x0001	<i>PriceDecreaseRandomizeMinutes</i>	uint8	0x00 to 0x3C	RW	0x0F	O
0x0002	<i>CommodityType</i>	enum8	0x00 to 0xFF	R	-	O

13162 **10.2.3.2.1 PriceIncreaseRandomizeMinutes Attribute**

13163 The PriceIncreaseRandomizeMinutes attribute represents the maximum amount of time to be used when  
 13164 randomizing the response to a price increase. Note that although the granularity of the attribute is in minutes, it  
 13165 is recommended the granularity of the randomization used within a responding device be in seconds or smaller. If a  
 13166 device responds to a price increase it must choose a random amount of time, in seconds or smaller, between 0 and  
 13167 PriceIncreaseRandomizeMinutes minutes. The device must implement that random amount of time before or after  
 13168 the price change. How and if a device will respond to a price increase is up to the manufacturer. Whether to respond  
 13169 before or after the price increase is also up to the manufacturer.

13170 As an example, a water heater with a PriceIncreaseRandomizeMinutes set to 6 could choose to lower its set point  
 13171 315 seconds (but not more than 360 seconds) before the price increases.

13172 The valid range for this attribute is 0x00 to 0x3C.

13173 If PriceIncreaseRandomizeMinutes or PriceDecreaseRandomizeMinutes attributes are not supported by the client,  
 13174 then it should use the default values for the attributes as specified in the Price Client Cluster Attribute table.

13175 **10.2.3.2.2 PriceDecreaseRandomizeMinutes Attribute**

13176 The PriceDecreaseRandomizeMinutes attribute represents the maximum number of minutes to be used when  
 13177 randomizing the response to a price decrease. Note that although the granularity of the attribute is in minutes, it is  
 13178 recommended the granularity of the randomization used within a responding device be in seconds or smaller. If a  
 13179 device responds to a price decrease it must choose a random amount of time, in seconds or smaller, between 0 and  
 13180 PriceDecreaseRandomizeMinutes minutes and implement that random amount of time before or after the price  
 13181 change. How and if a device will respond to a price decrease is up to the manufacturer. Whether to respond before  
 13182 or after the price increase is also up to the manufacturer.

13183 As an example, a dishwasher with a PriceDecreaseRandomizeMinutes set to 15 could choose to start its wash  
 13184 cycle 723 seconds (but not more than 900 seconds) after the price decreases.

13185 The valid range for this attribute is 0x00 to 0x3C.

13186 **10.2.3.2.3 CommodityType Attribute**

13187 CommodityType provides a label for identifying the type of pricing client present. The attribute is an enumerated  
 13188 value representing the commodity. The defined values are represented by the non-mirrored values (0-127) in the  
 13189 MeteringDeviceType attribute enumerations (refer to Table 10-42).

13190 **10.2.3.3 Commands Received**

13191 The client receives the cluster-specific response commands detailed in sub-clause 10.2.2.4.

### 13192 **10.2.3.4 Commands Generated**

13193 The client generates the cluster-specific commands detailed in sub-clause 10.2.2.3, as required by the application.

## 13194 **10.2.4 Application Guidelines**

### 13195 **10.2.4.1 Registering for Commands**

13196 Devices should use bind request to register for unsolicited Publish Price, Display Message and Load Control  
13197 Event commands.

### 13198 **10.2.4.2 Attribute Reporting**

13199 Attribute reporting may be used for sending information in the Price Server Cluster Attributes table. The Price  
13200 Cluster attributes can be polled periodically for updates. Polling should not occur more frequently than  
13201 recommended in 10.4.3.2. Use of the Report Attribute command without report configuration may be used for  
13202 unsolicited notification of an attribute value change. Sleepy devices may have to poll.

### 13203 **10.2.4.3 Block Tariffs**

13204 Upon reaching the Start Time of a received Publish Price command, a device's behavior will depend on the  
13205 values of the Number of Block Thresholds and Number of Price Tiers fields. A client device needing to  
13206 determine if it should use Block Pricing shall send a Get Current Price command to the Price server and check the  
13207 Number of Block Thresholds in the Publish Price response. Any value between 1 and 15 indicates that Block Pricing  
13208 shall be used.

13209 The prices for a commodity being delivered to the premises shall be taken from the Block Pricing Information  
13210 Attribute Set whenever Block Pricing is active.

#### 13211 **10.2.4.3.1 TOU Charging Only**

13212 Indicated by the Number of Block Thresholds field being set to zero. Charging shall be according to the price  
13213 fields within the Publish Price command itself.

#### 13214 **10.2.4.3.2 Block Charging Only**

13215 Indicated by the Number of Price Tiers fields being set to zero while the Number of Block Thresholds is between  
13216 0x01 and 0x0F.

13217 A server shall not update the Block Threshold and Block Price attribute sets of an active Block Period. Updates to  
13218 these attribute sets can only be done by creating a new Block Period. The server may create a new active Block  
13219 Period by updating either Block Period Start Time (attribute StartOfBlockPeriod) alone or Block Period Duration in  
13220 Minutes (attribute BlockPeriodDuration) followed by Block Period Start Time (attribute StartOfBlockPeriod) along  
13221 with updating other attributes as desired.

13222 When a server transmits a Publish Price command it shall additionally fill fields necessary to support backwards  
13223 compatibility with clients that may not support Block Charging. The Price field shall be set according to the Block  
13224 Price Information Attribute Set. The Duration in Minutes field shall be set to 0xFFFF indicating the price is valid  
13225 “until changed”.

13226 A server shall additionally transmit a Publish Price command to clients under the following conditions:

- 13227 1. At the start of a Block Period
- 13228 2. When it is notified that a Block Threshold has been crossed

13229 3. When *Block Period Start Time* or *Block Period Duration in Minutes* have changed to indicate a new active  
13230 block period

13231 A client may cache attributes from the Block Threshold, Block Period, Block Price, and Billing Period attribute sets.  
13232 Cached attributes are valid only during the active Block Period when received. Upon reaching Block Period Start  
13233 Time or detecting a new active Block Period, the client should retrieve updated values for cached attributes.

13234 A client shall check for a new active Block Period on receipt of an asynchronous Publish Price command (i.e. not  
13235 required on a Publish Price command in response to Get Current Price) by checking Block Period Start Time and  
13236 Block Period Duration in Minutes for update. Additionally, it shall infrequently (e.g. once an hour) query the  
13237 StartOfBlockPeriod and BlockPeriodDuration attributes to verify that the Block Period has not ended early.

#### 13238 **10.2.4.3.3 Block/TOU Combination Charging**

13239 **Note:** The following application guidelines that pertain to Block/TOU Combination Charging in this revision of  
13240 this specification are provisional and not certifiable. This text may change before reaching certifiable status in a  
13241 future revision of this specification.

13242 The Number of Block Thresholds and Number of Price Tiers fields will both be set to non-zero values, indicating the  
13243 number of blocks and number of tiers respectively being used. The start of a Block period shall be indicated by the  
13244 value of the Block Period Start Time field within a Publish Block Period command. Upon reaching the Block  
13245 Period Start Time, the attributes for the required number of Block Thresholds, together with the Block Prices for all  
13246 required blocks for the selected tier should be fetched from the server. The Block Period Duration in Minutes  
13247 field shall indicate the length of the block period.

13248 A Publish Price command will be received for the start of each new TOU period during a block period. At this  
13249 point the attributes for the Block Prices for all required blocks for the newly activated tier should be fetched from  
13250 the server.

#### 13251 **10.2.4.3.4 Application Guidelines for Block Pricing Under Specific Events**

13252 HAN device not communicating with meter for extended period of time:

13253 In this situation, when the HAN device reconnects with the meter, it will need to read the Block Information Set to  
13254 calculate the correct cost for the given period. This is done by applying the prices for each block/tier combination  
13255 to the consumption information for each block/tier combination. If a block period has passed while the HAN  
13256 device was not communicating with the meter, then the prior period consumption information will not be known  
13257 and the prior period cost cannot be calculated by the HAN device.

13258 Meter installation or swap-out:

13259 The new meter will need to be configured with the appropriate block thresholds, pricing, and block duration by  
13260 the utility. If this does not occur precisely at the start of that customer's billing period, the utility will need to (a)  
13261 pro-rate these amounts over the remaining billing period duration and (b) decide how to handle the initial portion of  
13262 the period. Any information from the initial part of the billing period will be lost when the new meter is installed.  
13263 As such, HAN devices may not display accurate information for this billing period and utilities should advise  
13264 customers of this situation. As a typical meter lifetime is expected to be in the range of 10 to 20 years, this event  
13265 is expected to be rare.

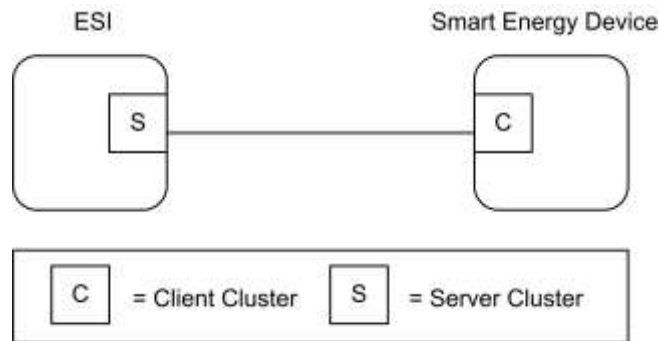
13266 **10.3 Demand Response and Load Control**

13267 **10.3.1 Overview**

13268 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
13269 etc.

13270 This cluster provides an interface to the functionality of Smart Energy Demand Response and Load Control. Devices  
13271 targeted by this cluster include thermostats and devices that support load control.

13272 **Figure 10-13. Demand Response/Load Control Cluster Client Server Example**



*Note: Device names are examples for illustration purposes only*

13273  
13274

13275 Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream demand response/load  
13276 control management systems and subsequent data stores.

13277 **10.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

13278 **10.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	DRLC	Type 1 (client to server)

13279 **10.3.1.3 Cluster Identifiers**

Identifier	Name
0x0701	Demand Response and Load Control

13280 **10.3.2 Server**

13281 By default the ESI will be labeled as the Server side in the cluster descriptions, being able to initiate load control  
13282 commands to other devices in the network.

13283 **10.3.2.1 Dependencies**

- 13284 A server device shall be capable of storing at least two load control events.
- 13285 Events carried using this cluster include a timestamp with the assumption that target devices maintain a real-time clock. Devices can acquire and synchronize their internal clocks with the ESI as described in sub-clause 3.12.
- 13286
- 13287 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.
- 13288
- 13289 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.
- 13290

13291 **10.3.2.2 Attributes**

- 13292 There are no attributes for the Demand Response and Load Control Cluster server.

13293 **10.3.2.3 Commands Generated**

- 13294 The command IDs generated by the Demand Response and Load Control cluster server are listed in Table 10-18.

13295 **Table 10-18. Command IDs for the Demand Response and Load Control Server**

Command Identifier	Description	M/O
0x00	Load Control Event	M
0x01	Cancel Load Control Event	M
0x02	Cancel All Load Control Events	M

13296 **10.3.2.3.1 Load Control Event Command**

13297 **10.3.2.3.1.1 Payload Format**

- 13298 The Load Control Event command payload shall be formatted as illustrated in Figure 10-14.

13299 **Figure 10-14. Format of the Load Control Event Command Payload**

Octets	4	2	1	4	2	1	1
Data Type	uint32	map16	uint8	UTC	uint16	uint8	uint8
Field Name	Issuer Event ID (M)	Device Class (M)	Utility Enrollment Group (M)	Start Time (M)	Duration in Minutes (M)	Criticality Level (M)	Cooling Temperature Offset (O)

13300

<b>Octets</b>	1	2	2	1	1	1
<b>Data Type</b>	uint8	int16	int16	int8	uint8	map8
<b>Field Name</b>	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)

13301 **Note:** M = Mandatory field, O = Optional field. **All fields must be present in the payload.** Optional fields will be  
13302 marked with specific values to indicate they are not being used.

### 13303 10.3.2.3.1.1.1 Payload Details

13304 **Issuer Event ID (mandatory):** Unique identifier generated by the Energy provider. The value of this field allows  
13305 matching of Event reports with a specific Demand Response and Load Control event. The expected value contained  
13306 in this field shall be a unique number managed by upstream systems or a UTC based time stamp (UTC data type)  
13307 identifying when the Load Control Event was issued.

13308 **Device Class (mandatory):** Bit encoded field representing the Device Class to apply the current Load Control  
13309 Event. Each bit, if set individually or in combination, indicates the class device(s) needing to participate in the event.  
13310 (Note that the participating device may be different than the controlling device. For instance, a thermostat may act  
13311 on behalf of an HVAC compressor or furnace and/or Strip Heat/Baseboard Heater and should take action on their  
13312 behalf, as the thermostat itself is not subject to load shed but controls devices that are subject to load shed.) The  
13313 encoding of this field is in Table 10-19.

13314 **Table 10-19. Device Class Field BitMap/Encoding**

Bit	Description
0	HVAC Compressor or Furnace
1	Strip Heaters/Baseboard Heaters
2	Water Heater
3	Pool Pump/Spa/Jacuzzi
4	Smart Appliances
5	Irrigation Pump
6	Managed Commercial & Industrial (C&I) loads
7	Simple misc. (Residential On/Off) loads
8	Exterior Lighting
9	Interior Lighting
10	Electric Vehicle
11	Generation Systems

13315  
13316 Device manufacturers shall recognize the Device Class or set of Devices Classes that corresponds to its  
13317 functionality. For example, a thermostat (PCT) may react when Bit 0 is set since it controls the HVAC and/or  
13318 furnace. Another example is a device that acts like an EMS where it controls exterior lights, interior lights, and  
13319 simple misc. load control devices. In this case the EMS would react when Bits 7, 8, or 9 are set individually or in  
13320 combination.

13321 **Utility Enrollment Group (mandatory):** The Utility Enrollment Group field can be used in conjunction with the  
13322 Device Class bits. It provides a mechanism to direct Load Control Events to groups of Devices. Example, by



13323 assigning two different groups relating to either Demand Response programs or geographic areas, Load Control  
 13324 Events can be further directed for a sub-set of Device Classes (i.e. Device Class Bit 0 and Utility Enrollment Group  
 13325 #1 vs. Device Class Bit0 and Utility Enrollment Group #2). 0x00 addresses all groups, and values 0x01 to 0xFF  
 13326 address individual groups that match. Please refer to sub-clause 10.3.3.2.1 for further details.

13327 If the Device Class and/or Utility Enrollment Group fields don't apply to your End Device, the Load Control Event  
 13328 command shall be ignored by either dropping the message and not replying at all or by sending back a Default  
 13329 Response message with a SUCCESS status code.

13330 **Start Time (mandatory):** UTC Timestamp representing when the event is scheduled to start. A start time of  
 13331 0x00000000 is a special time denoting "now." If the device would send an event with a Start Time of now, adjust  
 13332 the Duration In Minutes field to correspond to the remainder of the event.

13333 **Duration In Minutes (mandatory):** Duration of this event in number of minutes. Maximum value is 1440 (one day).

13334 **Criticality Level (mandatory):** This field defines the level of criticality of this event. The action taken by load  
 13335 control devices for an event can be solely based on this value, or combination with other Load Control Event  
 13336 fields supported by this device. For example, additional fields such as Average Load Adjustment Percentage,  
 13337 Duty Cycle, Cooling Temperature Offset, Heating Temperature Offset, Cooling Temperature Set Point or Heating  
 13338 Temperature Set Point can be used in combination with the Criticality level. Criticality levels are listed in  
 13339 Table 10-20.

13340 **Table 10-20. Criticality Levels**

Criticality Level	Level Description	Participation
1	Green	Voluntary
2	1	Voluntary
3	2	Voluntary
4	3	Voluntary
5	4	Voluntary
6	5	Voluntary
7	Emergency	Mandatory
8	Planned Outage	Mandatory
9	Service Disconnect	Mandatory
0x0A to 0x0F	Utility Defined	Utility Defined

13341

13342 The criticality level 0x0 and 0x10 to 0xFF are reserved for future profile changes and not used.

13343 "Green" event, level 0x01, may be used to denote that the energy delivered uses an abnormal amount from non-  
 13344 "green" sources. Participation in this event is voluntary.

13345 The criticality levels 0x02 through 0x06 (Levels 1 through 5) indicate progressively increasing levels of load  
 13346 reduction are being requested by the utility. Participation in these events is voluntary.

13347 The criticality level 0x07 is used to indicate an "Emergency" event. Participation in this event is mandatory, as  
 13348 defined by the utility. The expected response to this event is termination of all non-essential energy use, as defined  
 13349 by the utility. Exceptions to participation in this event type must be managed by the utility.

13350 The criticality level 0x08 is used to indicate a "Planned Outage" event. Participation in this event is mandatory, as  
 13351 defined by the utility. The expected response to this event is termination of delivery of all non-essential energy, as  
 13352 defined by the utility. Exceptions to participation in this event type must be managed by the utility.

- 13353 The criticality level 0x09 is used to indicate a “Service Disconnect” event. Participation in this event is mandatory,  
13354 as defined by the utility. The expected response to this event is termination of delivery of all non-essential energy, as  
13355 defined by the utility. Exceptions to participation in this event type must be managed by the utility.
- 13356 Levels 0x0A to 0x0F are available for Utility Defined criticality levels.
- 13357 **Cooling Temperature Offset (optional):** Requested offset to apply to the normal cooling setpoint at the time of the  
13358 start of the event in + 0.1 °C.
- 13359 **Heating Temperature Offset (optional):** Requested offset to apply to the normal heating setpoint at the time of the  
13360 start of the event in + 0.1 °C.
- 13361 The Cooling and Heating Temperature Offsets represent a temperature change (Delta Temperature) that will be  
13362 applied to both the associated heating and cooling set points. The temperature offsets (Delta Temperatures) will be  
13363 calculated per the Local Temperature in the Thermostat. The calculated temperature will be interpreted as the  
13364 number of degrees to be added to the cooling set point and subtracted from the heating set point. Sequential demand  
13365 response events are not cumulative. The Offset shall be applied to the normal setpoint.
- 13366 Each offset represents the temperature offset (Delta Temperature) in degrees Celsius, as follows: Delta Temperature  
13367 Offset / 10 = delta temperature in degrees Celsius. Where 0.00°C <= temperature <= 25.4 °C, corresponding to a  
13368 Temperature in the range 0x00 to 0x0FE. The maximum resolution this format allowed is 0.1 °C.
- 13369 A DeltaTemperature of 0xFF indicates that the temperature offset is not used.
- 13370 If a temperature offset is sent that causes the heating or cooling temperature set point to exceed the limit boundaries  
13371 that are programmed into the thermostat, the thermostat should respond by setting the temperature at the limit.
- 13372 **Cooling Temperature Set Point (optional):** Requested cooling set point in 0.01 degrees Celsius.
- 13373 **Heating Temperature Set Point (optional):** Requested heating set point in 0.01 degrees Celsius.
- 13374 Cooling and heating temperature set points will be defined and calculated per the LocalTemperature attribute in the  
13375 Thermostat Cluster (see Chapter 6).
- 13376 These fields represent the temperature in degrees Celsius, as follows:
- 13377 Cooling Temperature Set Point / 100 = temperature in degrees Celsius
- 13378 where -273.15°C <= temperature <= 327.67°C, corresponding to a Cooling and/or Heating Temperature Set Point in  
13379 the range 0x954d to 0x7fff
- 13380 The maximum resolution this format allows is 0.01°C.
- 13381 A Cooling or Heating Temperature Set Point of 0x8000 indicates that the temperature set point is not used.
- 13382 If a temperature is sent that exceeds the temperature limit boundaries that are programmed into the thermostat, the  
13383 thermostat should respond by setting the temperature at the limit.
- 13384 The thermostat shall not use a Cooling or Heating Temperature Set Point that causes the device to use more energy  
13385 than the normal setting.
- 13386 When both a Temperature Offset and a Temperature Set Point are provided, the thermostat may use either as defined  
13387 by the device manufacturer. The thermostat should use the setting that provides the lowest energy consumption.
- 13388 **Average Load Adjustment Percentage (optional):** Defines a maximum energy usage limit as a percentage of the  
13389 client implementations specific average energy usage. The load adjustment percentage is added to 100% creating a  
13390 percentage limit applied to the client implementations specific average energy usage. A -10% load adjustment  
13391 percentage will establish an energy usage limit equal to 90% of the client implementations specific average energy  
13392 usage. Each load adjustment percentage is referenced to the client implementations specific average energy usage.  
13393 There are no cumulative effects.
- 13394 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%  
13395 value will limit the energy usage to the client implementation’s specific average energy usage. A +100% value will  
13396 limit the energy usage to double the client implementation’s specific average energy usage.
- 13397 A value of 0x80 indicates the field is not used. All other values are reserved for future use.

13398 **Duty Cycle (optional):** Defines the maximum On state duty cycle as a percentage of time. Example, if the value is  
 13399 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  
 13400 100. A value of 0xFF indicates the field is not used. All other values are reserved for future use.

13401 Duty cycle control is a device specific issue and shall be managed by the device manufacturer. It is expected that the  
 13402 duty cycle of the device under control will span the shortest practical time period in accordance with the nature of  
 13403 the device under control and the intent of the request for demand reduction. For typical Device Classes, three  
 13404 minutes for each 10% of duty cycle is recommended. It is expected that the “off state” will precede the “on state”.

13405 **Event Control (mandatory):** Identifies additional control options for the event. The BitMap for this field is  
 13406 described in Table 10-21.

13407 **Table 10-21. Event Control Field BitMap**

Bit	Description
0	1= Randomize Start time, 0=Randomized Start not Applied
1	1= Randomize End time, 0=Randomized End not Applied

13408  
 13409 **Note:** The randomization attribute will be used in combination with two bits to determine if the Event Start and Stop  
 13410 Times are randomized. By default devices will randomize the start and stop of an event. Refer to sub-clause  
 13411 10.3.3.2.2 and sub-clause 10.3.3.2.3 for the settings of these values.

13412 **10.3.2.3.1.1.2 When Generated**

13413 This command is generated when the ESI wants to control one or more load control devices, usually as the result of  
 13414 an energy curtailment command from the Smart Energy network.

13415 **10.3.2.3.1.1.3 Responses to Load Control Event**

13416 The server receives the cluster-specific commands detailed in sub-clause 10.3.3.3.1.

13417 **10.3.2.3.2 Cancel Load Control Event Command**

13418 **10.3.2.3.2.1 Payload Format**

13419 The Cancel Load Control Event command payload shall be formatted as illustrated in Figure 10-15.

13420 **Figure 10-15. Format of the Cancel Load Control Event Payload**

<b>Octets</b>	4	2	1	1	4
<b>Data Type</b>	uint32	map16	uint8	map8	UTC
<b>Field Name</b>	Issuer Event ID	Device Class (M)	Utility Enrollment Group (M)	Cancel Control (M)	Effective Time (M)

13421 **10.3.2.3.2.1.1 Payload Details**

13422 **Issuer Event ID (mandatory):** Unique identifier generated by the Energy provider. The value of this field allows  
 13423 matching of Event reports with a specific Demand Response and Load Control event. It's expected the value  
 13424 contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data  
 13425 type) identifying when the Load Control Event was issued.

13426 **Device Class (mandatory):** Bit encoded field representing the Device Class to apply the current Load Control  
 13427 Event. Each bit, if set individually or in combination, indicates the class device(s) needing to participate in the event.  
 13428 (Note that the participating device may be different than the controlling device. For instance, a thermostat may act  
 13429 on behalf of an HVAC compressor or furnace and/or Strip Heat/Baseboard Heater and should take action on their

- 13430 behalf, as the thermostat itself is not subject to load shed but controls devices that are subject to load shed.) The  
13431 encoding of the Device Class is listed in Table 10-19.
- 13432 **Utility Enrollment Group (mandatory):** The Utility Enrollment Group field can be used in conjunction with the  
13433 Device Class bits. It provides a mechanism to direct Load Control Events to groups of Devices. Example, by  
13434 assigning two different groups relating to either Demand Response programs or geographic areas, Load Control  
13435 Events can be further directed for a sub-set of Device Classes (i.e. Device Class Bit 0 and Utility Enrollment Group  
13436 #1 vs. Device Class Bit0 and Utility Enrollment Group #2). 0x00 addresses all groups, and values 0x01 to 0xFF  
13437 address individual groups that match. Please refer to sub-clause 10.3.2.3.2.1 for further details.
- 13438 If the Device Class and/or Utility Enrollment Group fields don't apply to your End Device, the Cancel Load Control  
13439 Event command is ignored.
- 13440 Device Class and/or Utility Group fields must be the same for a Cancel Load Control Event command as they were  
13441 for the command to create the event. Should these fields be different there is no defined behavior for how DRLC  
13442 servers should maintain their tables for replying to Get Scheduled Events commands.
- 13443 **Cancel Control (mandatory):** The encoding of the Cancel Control is listed in Table 10-22.

13444 **Table 10-22. Cancel Control**

Bit	Description
0	To be used when the Event is currently in process and acted upon as specified by the Effective Time field of the Cancel Load Control Event command.  A value of Zero (0) indicates that randomization is overridden and the event should be terminated immediately at the Effective Time.  A value of One (1) indicates the event should end using randomization settings in the original event.

- 13445
- 13446 **Effective Time (mandatory):** UTC Timestamp representing when the canceling of the event is scheduled to start.  
13447 An effective time of 0x00000000 is a special time denoting “now.” If the device would send an event with an  
13448 Effective Time of now, adjust the Duration In Minutes field to correspond to the remainder of the event.
- 13449 **Note:** This field is deprecated; a Cancel Load Control command shall now take immediate effect. A value of  
13450 0x00000000 shall be used in all Cancel Load Control commands
- 13451 **10.3.2.3.2.1.2 When Generated**
- 13452 This command is generated when the ESI wants to cancel previously scheduled control of one or more load control  
13453 devices, usually as the result of an energy curtailment command from the Smart Energy network.
- 13454 **10.3.2.3.2.1.3 Responses to Cancel Load Control Event**
- 13455 The server receives the cluster-specific commands detailed in sub-clause 10.3.3.3.1.
- 13456 **Note:** If the Cancel Load Control Event command is received after the event has ended, the device shall reply using  
13457 the “Report Event Status Command” with an Event Status of “Rejected -Invalid Cancel Command (Undefined  
13458 Event)”.
- 13459 **10.3.2.3.3 Cancel All Load Control Events Command**
- 13460 **10.3.2.3.3.1 Payload Format**
- 13461 The Cancel All Load Control Events command payload shall be formatted as illustrated in Table 10-23.

13462 **Table 10-23. Format of the Cancel All Load Control Events Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	map8
<b>Field Name</b>	Cancel Control

13463 **10.3.2.3.3.1 Payload Details**

13464 **Cancel Control:** The encoding of the Cancel Control is listed in Table 10-24.

13465 **Table 10-24. Cancel All Command Cancel Control Field**

<b>Bit</b>	<b>Description</b>
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.

13466 **10.3.2.3.3.2 When Generated**

13467 This command is generated when the ESI wants to cancel all events for control device(s).

13468 **10.3.2.3.3.3 Responses to Cancel All Load Control Events**

13469 The server receives the cluster-specific commands detailed in sub-clause 10.3.3.1. The Cancel All Load Control  
 13470 Events command is processed by the device as if individual Cancel Load Control Event commands were received  
 13471 for all of the currently stored events in the device. The device will respond with a “Report Event Status Command”  
 13472 for each individual load control event canceled.

13473 **10.3.2.4 Commands Received**

13474 The server receives the cluster-specific commands detailed in sub-clause 10.3.3.

13475 **10.3.3 Client**

13476 This section identifies the attributes and commands provided by Client devices.

13477 **10.3.3.1 Dependencies**

13478 Devices receiving and acting upon Load Control Event commands must be capable of storing and supporting at  
 13479 least three unique instances of events. As a highly recommended recovery mechanism, when maximum storage of  
 13480 events has been reached and additional Load Control Events are received that are unique (not superseding currently  
 13481 stored events), devices should ignore additional Load Control Events and when storage becomes available, utilize  
 13482 the GetScheduledEvents command to retrieve any previously ignored events.

13483 Events carried using this cluster include a timestamp with the assumption that target devices maintain a real  
 13484 time clock. Devices can acquire and synchronize their internal clocks with the ESI as described in the Time cluster  
 13485 sub-clause 3.12.

13486 Devices MAY ‘drop’ events received before they have received and resolved time (‘dropping’ an event is defined as  
 13487 sending a default response with status code SUCCESS).

13488 If a device does not support a real time clock, it's assumed the device will ignore all values within the Time field  
 13489 except the “Start Now” value.

13490 Additionally, for devices without a real time clock it's assumed those devices will utilize a method (i.e. ticks,  
13491 countdowns, etc.) to approximate the correct duration period.

### 13492 **10.3.3.2 Client Cluster Attributes**

13493 **Table 10-25. Demand Response Client Cluster Attributes**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Default</b>	<b>M/O</b>
0x0000	<i>UtilityEnrollmentGroup</i>	uint8	0x00 to 0xFF	RW	0x00	M
0x0001	<i>StartRandomizeMinutes</i>	uint8	0x00 to 0x3C	RW	0x1E	M
0x0002	<i>StopRandomizeMinutes</i>	uint8	0x00 to 0x3C	RW	0x1E	M
0x0003	<i>DeviceClassValue</i>	uint16	0x0000 to 0xFFFF	RW	-	M

#### 13494 **10.3.3.2.1 UtilityEnrollmentGroup Attribute**

13495 The UtilityEnrollmentGroup provides a method for utilities to assign devices to groups. In other words, Utility  
13496 defined groups provide a mechanism to arbitrarily group together different sets of load control or demand response  
13497 devices for use as part of a larger utility program. The definition of the groups, implied usage, and their assigned  
13498 values are dictated by the Utilities and subsequently used at their discretion, therefore outside the scope of this  
13499 specification. The valid range for this attribute is 0x00 to 0xFF, where 0x00 (the default value) indicates the device  
13500 is a member of all groups and values 0x01 to 0xFF indicates that the device is member of that specified group.

#### 13501 **10.3.3.2.2 StartRandomizationMinutes Attribute**

13502 The StartRandomizedMinutes represents the maximum number of minutes to be used when randomizing the start  
13503 of an event. As an example, if StartRandomizedMinutes is set for 3 minutes, the device could randomly select 2  
13504 minutes (but never greater than the 3 minutes) for this event, causing the start of the event to be delayed by two  
13505 minutes. The valid range for this attribute is 0x00 to 0x3C where 0x00 indicates start event randomization is not  
13506 performed.

#### 13507 **10.3.3.2.3 EndRandomizationMinutes Attribute**

13508 The EndRandomizedMinutes represents the maximum number of minutes to be used when randomizing the end  
13509 of an event. As an example, if EndRandomizedMinutes is set for 3 minutes, the device could randomly select one  
13510 minute (but never greater than 3 minutes) for this event, causing the end of the event to be delayed by one  
13511 minute. The valid range for this attribute is 0x00 to 0x3C where 0x00 indicates end event randomization is not  
13512 performed.

#### 13513 **10.3.3.2.4 DeviceClassValue Attribute**

13514 The DeviceClassValue attribute identifies which bits the device will match in the Device Class fields. Please refer  
13515 to Table 10-19 for further details. Although the attribute has a RW access property, the device is permitted to  
13516 refuse to change the DeviceClass by setting the status field of the corresponding write attribute status record to  
13517 NOT\_AUTHORIZED.

13518 Although, for backwards compatibility, the Type cannot be changed, this 16-bit integer should be treated as if it  
13519 were a 16-bit bitmap.

13520 Device Class and/or Utility Enrollment Group fields are to be used as filters for deciding to accept or ignore a Load  
13521 Control Event or a Cancel Load Control Event command. There is no requirement for a device to store or remember  
13522 the Device Class and/or Utility Enrollment Group once the decision to accept the event has been made. A

13523 consequence of this is that devices that accept multiple device classes may have an event created for one device  
 13524 class superseded by an event created for another device class.

13525 In-Home Displays should report the device classes that they are interested in. An IHD that wishes to display all  
 13526 possible Load Control Events, even for classes not yet defined, should indicate a device class of 0xFFFF; this will  
 13527 allow DRLC servers to optimize the number of DRLC events they unicast, such that they are only sent to those  
 13528 devices that are interested in them.

### 10.3.3.3 Commands Generated

13529 The command IDs generated by the Demand Response and Load Control client cluster are listed in Table 10-26.

13531 **Table 10-26. Generated Command IDs for the Demand Response and Load Control Client**

Command Identifier Field Value	Description	M/O
0x00	Report Event Status	M
0x01	Get Scheduled Events	M

#### 10.3.3.3.1 Report Event Status Command

##### 10.3.3.3.1.1 Payload Format

13533 The Report Event Status command payload shall be formatted as illustrated in Figure 10-16.  
 13534

13535 **Figure 10-16. Format of the Report Event Status Command Payload**

Octets	4	1	4	1	2	2
Data Type	uint32	uint8	UTC	uint8	uint16	uint16
Field Name	Issuer Event ID (M)	Event Status (M)	Event Status Time (M)	Criticality Level Applied (M)	Cooling Temperature Set Point Applied (O)	Heating Temperature Set Point Applied (O)

Octets	1		1	1	1	42
Data Type	int8		uint8	map8	uint8	opaque
Field Name	Average Load Adjustment Percentage Applied (O)		Duty Cycle Applied (O)	Event Control (M)	Signature Type (M)	Signature (O)

##### 10.3.3.3.1.1.1 Payload Details

13537 **Issuer Event ID (mandatory):** Unique identifier generated by the Energy provider. The value of this field allows  
 13538 matching of Event reports with a specific Demand Response and Load Control event. It's expected the value  
 13539 contained in this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data  
 13540 type) identifying when the Load Control Event was issued.  
 13541

13542 **Event Status (mandatory):** Table 10-27 lists the valid values returned in the Event Status field.

13543

Table 10-27. Event Status Field Values

Value	Description
0x01	Load Control Event command received
0x02	Event started
0x03	Event completed
0x04	User has chosen to “Opt-Out”, user will not participate in this event
0x05	User has chosen to “Opt-In”, user will participate in this event
0x06	The event has been cancelled
0x07	The event has been superseded
0x08	Event partially completed with User “Opt-Out”
0x09	Event partially completed due to User “Opt-In”
0x0A	Event completed, no User participation (Previous “Opt-Out”)
0xF8	Rejected -Invalid Cancel Command (Default)
0xF9	Rejected -Invalid Cancel Command (Invalid Effective Time)
0xFB	Rejected -Event was received after it had expired (Current Time > Start Time + Duration)
0xFD	Rejected -Invalid Cancel Command (Undefined Event)
0xFE	Load Control Event command Rejected

13544

13545 Should a device issue one or more “OptOut” or “OptIn” RES commands during an event that is eventually  
13546 cancelled, the event shall be recorded as a cancelled event (Status = 0x06) at its effective time.

13547 Should a device issue one or more “OptOut” or “OptIn” RES commands during an event that is not cancelled, the  
13548 event shall be recorded as partially completed based on the last RES command sent (Status = 0x08 or 0x09).

13549 When a device returns a status of 0xFD (Rejected -Invalid Cancel Command (Undefined Event)), all optional  
13550 fields should report their “Ignore” values.

13551 When a device receives a duplicate RES command, it should ignore the duplicate commands. Please note: As a  
13552 recommended best practice, ESI applications should provide a mechanism to assist in filtering duplicate messages  
13553 received on the WAN.

13554 **Event Status Time (mandatory):** UTC Timestamp representing when the event status occurred. This field shall  
13555 not use the value of 0x00000000.

13556 **Criticality Level Applied (mandatory):** Criticality Level value applied by the device, see the corresponding  
13557 field in the Load Control Event command for more information.

13558 **Cooling Temperature Set Point Applied (optional):** Cooling Temperature Set Point value applied by the device,  
13559 see the corresponding field in the Load Control Event command for more information. The value 0x8000 means that  
13560 this field has not been used by the end device.

13561 **Heating Temperature Set Point Applied (optional):** Heating Temperature Set Point value applied by the device,  
13562 see the corresponding field in the Load Control Event command for more information. The value 0x8000 means that  
13563 this field has not been used by the end device.

13564 **Average Load Adjustment Percentage Applied (optional):** Average Load Adjustment Percentage value applied by  
13565 the device, see the corresponding field in the Load Control Event command for more information. The value 0x80  
13566 means that this field has not been used by the end device.



13567 **Duty Cycle Applied (optional):** Defines the maximum On state duty cycle applied by the device. The value  
 13568 0xFF means that this field has not been used by the end device. Refer to sub-clause 10.3.2.3.1.1.1.

13569 **Event Control (mandatory):** Identifies additional control options for the event. Refer to sub-clause 10.3.2.3.1.1.1.

13570 **Signature Type (mandatory):** An 8-bit Unsigned integer enumerating the type of algorithm use to create the  
 13571 Signature. The enumerated values are shown in Table 10-28:

13572 **Table 10-28. Enumerated Values of Signature Types**

Enumerated Value	Signature Type
0x00	No Signature
0x01	ECDSA

13573

13574 If the signature field is not used, the signature type shall be set to 0x00, which will be used to indicate “no  
 13575 signature.” The signature field shall be filled with (48) 0xFF values.

13576 **Signature (optional) :** A non-repudiation signature created by using the Matyas-Meyer-Oseas hash function  
 13577 (specified in Annex B.6 in [Z1]) used in conjunction with ECDSA. The signature creation process will occur in  
 13578 two steps:

1. Pass the first ten fields, which includes all fields up to the Signature field, of the Report Event Status command (listed in Figure 10-16) through ECDSA using the device's ECC Private Key, generating the signature (r,s).

13581 **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.

13584 **Note:** the lengths of r and s are implicit, based on the curve used. Verifying the signature will require  
 13585 breaking the signature field back into the discrete components r and s, based on the length.

13586 **10.3.3.3.1.2 When Generated**

13587 This command is generated when the client device detects a change of state for an active Load Control event. (The  
 13588 transmission of this command should be delayed after a random delay between 0 and 5 seconds, to avoid a potential  
 13589 storm of packets.)

13590 **10.3.3.3.2 Get Scheduled Events Command**

13591 **Note:** The handling of this command is currently under review, and is likely to change in the next revision of the  
 13592 specification. Refer to CCB 1297 (and associated document 12-0180-00) for further information.

13593 This command is used to request that all scheduled Load Control Events, starting at or after the supplied Start Time, are  
 13594 re-issued to the requesting device. When received by the Server, one or more Load Control Event commands (see  
 13595 sub-clause 10.3.2.3.1) will be sent covering both active and scheduled Load Control Events.

13596 **10.3.3.3.2.1 Payload Format**

13597 The Get Scheduled Events command payload shall be formatted as illustrated in Figure 10-17.

13598 **Figure 10-17. Format of the Get Scheduled Events Command Payload**

<b>Octets</b>	4	1
<b>Data Type</b>	UTC	uint8
<b>Field Name</b>	Start Time (M)	Number of Events (M)

13599

13600 **Start Time (mandatory):** UTC Timestamp representing the minimum ending time for any scheduled or  
13601 currently active events to be resent. If either command has a Start Time of 0x00000000, replace that Start Time  
13602 with the current time stamp.

13603 **Number of Events (mandatory):** Represents the maximum number of events to be sent. A value of 0 would  
13604 indicate all available events are to be returned. Example: Number of Events = 1 would return the first event  
13605 with an EndTime greater than or equal to the value of Start Time field in the Get Scheduled Events command  
13606 (EndTime would be StartTime plus Duration of the event listed in the device's event table).

#### 13607 **10.3.3.3.2.2 When Generated**

13608 This command is generated when the client device wishes to verify the available Load Control Events or after a  
13609 loss of power/reset occurs and the client device needs to recover currently active or scheduled Load Control  
13610 Events.

13611 A ZCL Default Response with status NOT\_FOUND shall be returned when there are no events available.

### 13612 **10.3.3.4 Commands Received**

13613 The client receives the cluster-specific commands detailed in sub-clause 10.3.1.1.

### 13614 **10.3.3.5 Attribute Reporting**

13615 Attribute reporting is not expected to be used for this cluster. The Client side attributes are not expected to be  
13616 changed by the Client, only used during Client operations.

## 13617 **10.3.4 Application Guidelines**

13618 The criticality level is sent by the utility to the load control device to indicate how much load reduction is requested.  
13619 The utility is not required to use all of the criticality levels that are described in this specification. A load  
13620 control device is not required to provide a unique response to each criticality level that it may receive.

13621 The Average Load Adjustment Percentage, temperature offsets, and temperature set points are used by load control  
13622 devices and energy management systems on a “voluntary” or “optional” basis. These devices are not required to  
13623 use the values that are provided by the utility. They are provided as a recommendation by the utility.

13624 The load control device shall, in a manner that is consistent with this specification, accurately report event  
13625 participation by way of the Report Event Status message.

13626 The Average Load Adjustment Percentage is sent by the utility to the load control device to indicate how much load  
13627 reduction is requested. The load control device may respond to this information in a unique manner as defined by  
13628 the device manufacturer.

13629 The Duty Cycle is sent by the utility to the load control device to indicate the maximum “On state” for a device.  
13630 The control device may respond to this information in a unique manner as defined by the device manufacturer.

13631 The cooling temperature offset may be sent by the utility to the load shed control to indicate how much indoor  
13632 cooling temperature offset is requested. Response of a load control device to this information is not mandatory. The  
13633 control device may respond to this information in a unique manner as defined by the device manufacturer.

13634 The heating temperature offset may be sent by the utility to the load control device to indicate how much indoor  
13635 heating temperature offset is requested. The control device may respond to this information in a unique manner  
13636 as defined by the device manufacturer.

13637 The cooling temperature may be sent by the utility to the load control device to indicate the indoor cooling  
13638 temperature setting that is requested. The control device may respond to this information in a unique manner  
13639 as defined by the device manufacturer.

13640 The heating temperature may be sent by the utility to the load control device to indicate the indoor heating  
13641 temperature setting that is requested. The control device may respond to this information in a unique manner  
13642 as defined by the device manufacturer.

13643 **Note:** The most recent Load Control Event supersedes any previous Load Control Event command for the set of  
13644 Device Classes and groups for a given time. Nested events and overlapping events are not allowed. The current  
13645 active event will be terminated if a new event is started.

### 13646 **10.3.4.1 Load Control Rules, Server**

#### 13647 **10.3.4.1.1 Load Control Server, Identifying Use of SetPoint and Offset Fields**

13648 The use of the fields, Heating and Cooling Temperature Set Points and Heating and Cooling Temperature Offsets is  
13649 optional. All fields in the payload must be populated. Non-use of these fields by the Server is indicated by using the  
13650 following values: 0x8000 for Set Points and 0xFF for Offsets. When any of these four fields are indicated as  
13651 optional, they shall be ignored by the client.

#### 13652 **10.3.4.1.2 Load Control Server, Editing of Scheduled Events**

13653 Editing of a scheduled demand response event is not allowed. Editing of an active demand response event is not  
13654 allowed. Nested events and overlapping events are not allowed. The current active event will be terminated if a new  
13655 event is started.

### 13656 **10.3.4.2 Load Control Rules, Client**

#### 13657 **10.3.4.2.1 Start and Stop Randomization**

13658 When shedding loads (turning a load control device off), the load control device will optionally apply start time  
13659 randomization based on the values specified in the Event Control Bits and the Client's Start Randomization  
13660 Minutes attribute. By default, devices will apply a random delay as specified by the default values of start and  
13661 end randomization in the Demand Response Client Cluster Attributes table.

13662 When ending a load control event, the load control device will support the same randomization features as provided  
13663 in the start load control event.

#### 13664 **10.3.4.2.2 Editing of DR Control Parameters**

13665 In Load Control Device and energy management systems, editing of the demand response control parameters while  
13666 participating in an active demand response event is not allowed.

#### 13667 **10.3.4.2.3 Response to Price Events + Load Control Events**

13668 The residential system's response to price driven events will be considered in addition to the residential  
13669 system's response to demand response events. Demand response events which require that the residential system is  
13670 turned off have priority over price driven events. Demand response events which require that the residential system  
13671 go to a fixed setting point have priority over price driven events. In this case, the thermostat shall not use a  
13672 Cooling or Heating Temperature Set Point that causes the device to use more energy than the price driven event  
13673 setting.

#### 13674 **10.3.4.2.4 Opt-Out Messages**

13675 An event override message, "opt-out", will be sent by the load control device or energy management system if the  
13676 operator chooses not to participate in a demand response event by taking action to override the programmed

13677 demand reduction response. The override message will be sent at the start of the event. In the case where the  
13678 event has been acknowledged and started, the override message will be sent when the override occurs.

#### 13679 **10.3.4.2.5 Thermostat/HVAC Controls**

13680 A residential HVAC system will be allowed to change mode, from off to Heat, off to Cool, Cool to Heat, or Heat to  
13681 Cool, during a voluntary event which is currently active. The HVAC control must acknowledge the event, as if it  
13682 was operating, in that mode, at the start of the event. The HVAC control must obey the event rules that would have  
13683 been enforced if the system had been operating in that mode at the start of the active event.

13684 An event override message, “opt-out”, will be sent by the load control device or energy management system if the  
13685 operator chooses not to participate in a demand response event by taking action to override the programmed  
13686 demand reduction response. The override message will be sent at the start of the event. In the case where the  
13687 event has been acknowledged and started, the override message will be sent when the override occurs.

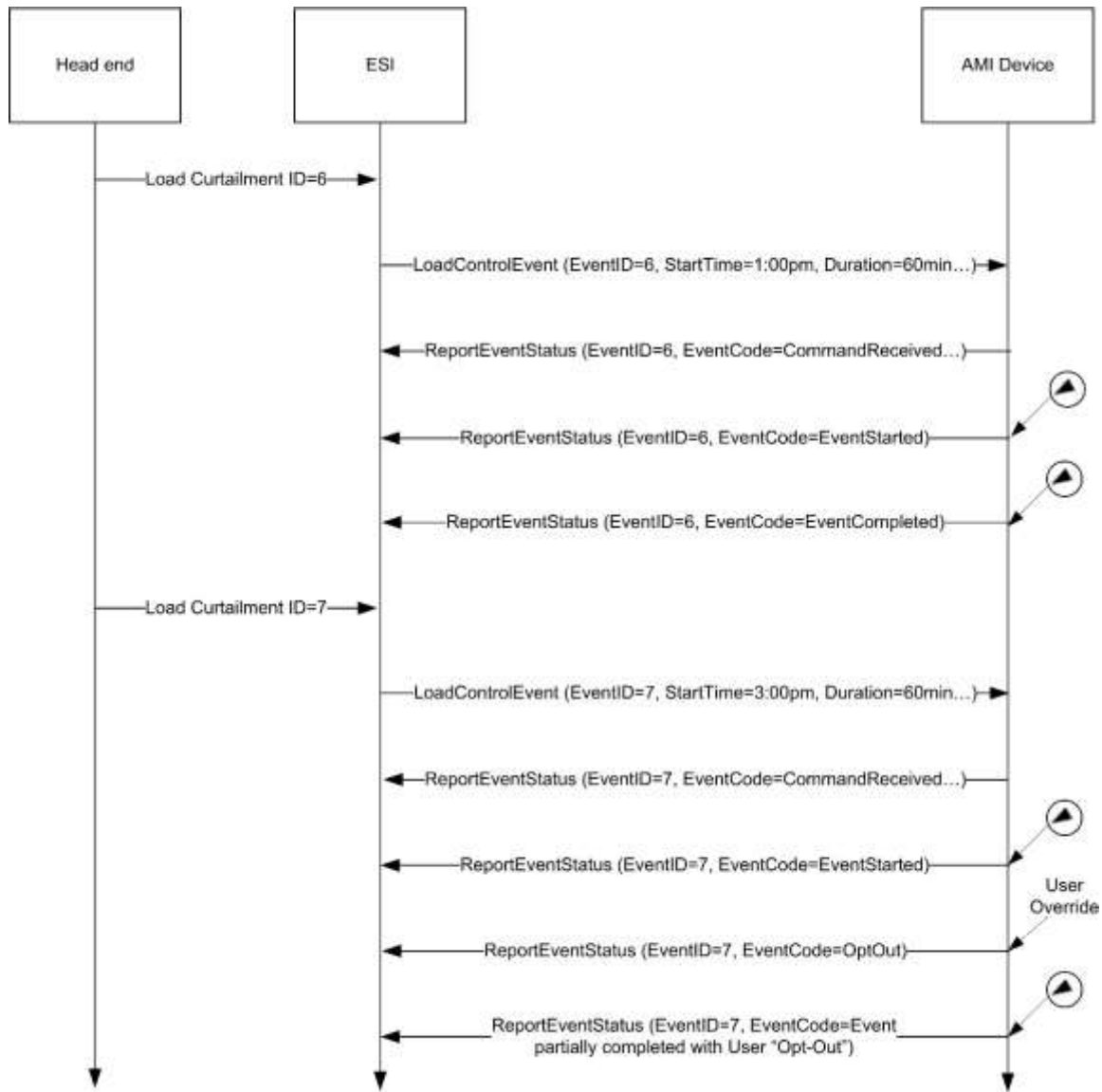
#### 13688 **10.3.4.2.6 Demand Response and Load Control Transaction Examples**

13689 The example in Figure 10-18 depicts the transactions that would take place for two events, one that is successful and  
13690 another that is overridden by the user.

13691

**Figure 10-18. Example of Both a Successful and an Overridden Load Curtailment Event**

13692



13693

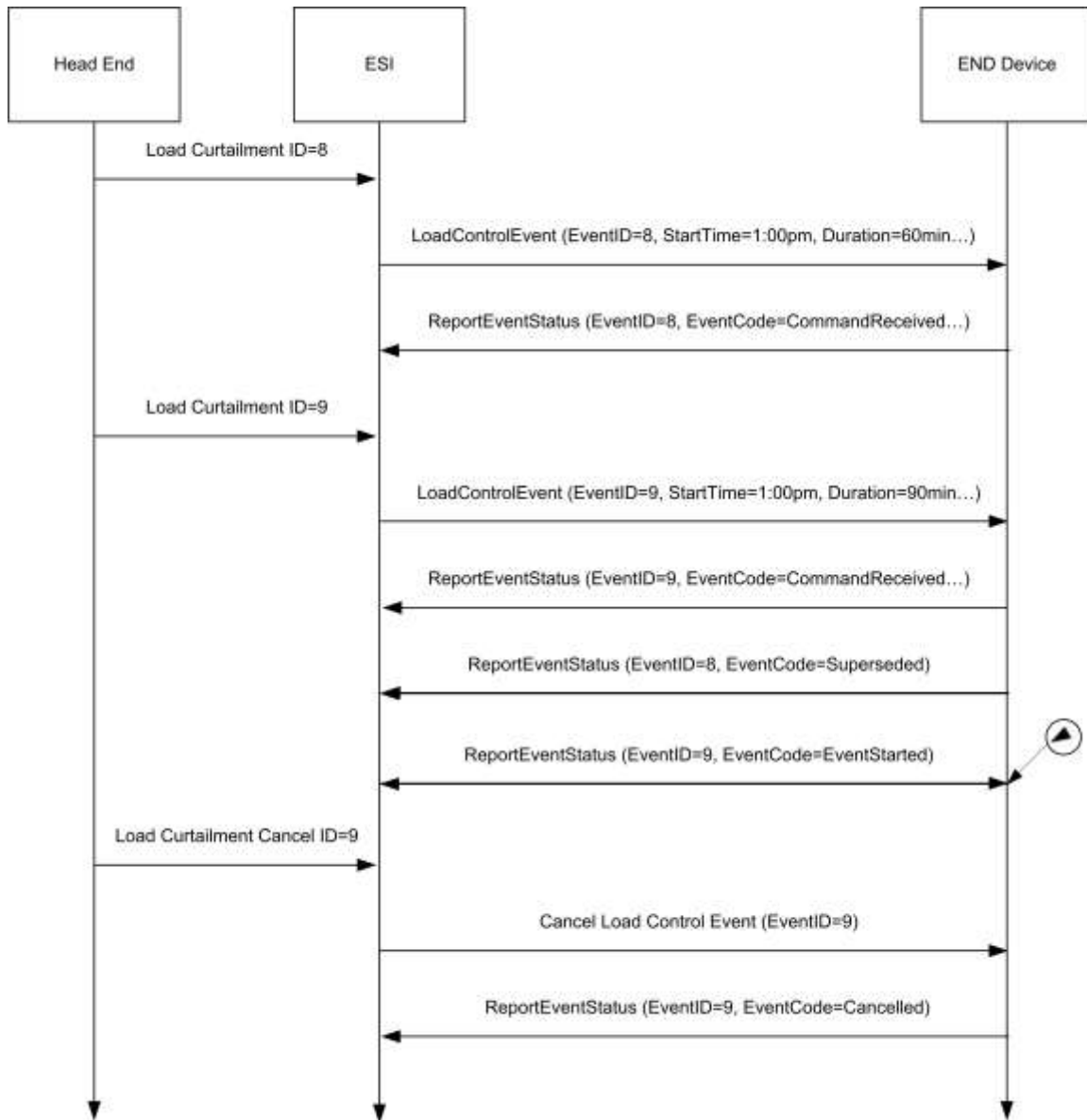
13694

13695

The example in Figure 10-19 depicts the transactions that would take place when an event is superseded by an event that is eventually cancelled.

13696

Figure 10-19. Example of a Load Curtailment Superseded and Another Cancelled



13697

13698

13699 Refer to section 10.3.5 for more information regarding the management and behavior of overlapping events.

### 13700 10.3.5 Rules and Guidelines for Overlapping Events

13701 This section describes multiple scenarios that Demand Response and Load Control devices may encounter over the  
 13702 Smart Energy network. The examples describe situations of overlapping events that are acceptable and where  
 13703 overlapping events that will be superseded due to conflicts.

#### 13704 10.3.5.1 Definitions

13705 **Start Time** – “Start Time” field contained within the Load Control Event packet indicating when the event should  
 13706 start. Please note, a “Start Time” value of 0x00000000 denotes “now” and the device should use its current time as  
 13707 the “Start Time.”

- 13708 **Duration** – “Duration” field contained within the Load Control Event packet indicating how long the event  
13709 should occur.
- 13710 **End Time** – Time when Event completes as calculated by adding Duration to Start Time.
- 13711 **Scheduled Period** – Represents the time between the Start Time and the End Time of the event.
- 13712 **Effective Start Time** -Represents time at which a specific device starts a load control event based on the Start  
13713 Time plus or minus any randomization offsets.
- 13714 **Effective End Time** – Represents time at which a specific device ends a load control event based on the Start  
13715 Time plus Duration, plus or minus any randomization offsets.
- 13716 **Effective Scheduled Period** – Represents the time between the Effective Start Time and the Effective End Time.
- 13717 **Overlapping Event** – Defined as an event where the Scheduled Period covers part or all of an existing, previously  
13718 scheduled event.
- 13719 **Successive Events** – Defined as two events where the scheduled End Time of the first event is equal the Start Time  
13720 of a subsequent scheduled event.
- 13721 **Nested Events** – Defined as two events where the scheduled Start Time and End Time of the second event falls  
13722 during the Scheduled Period of the first scheduled event and the second event is of shorter duration than the first  
13723 event.

### 13724 10.3.5.2 Rules and Guidelines

13725 The depicted behaviors and required application management decisions are driven from the following guidance and  
13726 rule set:

- 13727 1. Upstream Demand Response/Load Control systems and/or the ESI shall prevent mismanaged scheduling of  
13728 *Overlapping Events* or *Nested Events*. It is recognized Upstream Demand Response/Load Control systems  
13729 and/or the ESI will need to react to changing conditions on the grid by sending *Overlapping Events* or  
13730 *Nested Events* to supersede previous directives. But those systems must have the proper auditing and  
13731 management rules to prevent a cascading set of error conditions propagated by improperly scheduled events.
- 13732 2. When needed, Upstream Demand Response/Load Control systems and/or the ESI may resolve any event  
13733 scheduling conflicts by performing one of the following processes:
- 13734 a. Canceling individual events starting with the earliest scheduled event and re-issuing a new set of events.
- 13735 b. Canceling all scheduled events and re-issuing a new set of events.
- 13736 c. Sending *Overlapping Events* or *Nested Events* to supersede previous directives.
- 13737 It is recommended that process 2.c is used for most situations since it can allow a smoother change between  
13738 two sets of directives, but no way does it negate the responsibilities identified in rule #1.
- 13739 3. When an End Device receives an event with the *End Time* in the past (*End Time* < Current Time), this event  
13740 is ignored and a *Report Event Status* command is returned with the Event Status set to 0xFB (Rejected -  
13741 Event was received after it had expired).
- 13742 4. When an End Device receives an event with a Start Time in the past and an End Time in the future ((Start  
13743 Time < Current Time) AND (End Time > Current Time)), the event is processed immediately. The  
13744 Effective Start Time is calculated using the Current Time as the Start Time. Original End Time is  
13745 preserved.
- 13746 5. Regardless of the state of an event (scheduled or executing), when an End Device detects an Overlapping  
13747 Event condition the latest Overlapping Event will take precedence over the previous event. Depending on  
13748 the state of the event (scheduled or executing), one of the following steps shall take place:
- 13749 a. If the previous event is scheduled and not executing, the End Device returns a *Report Event Status*  
13750 command (referencing the previous event) with the Event Status set to 0x07 (The event has been

- 13751 superseded). After the *Report Event Status* command is successfully sent, the End Device can remove  
13752 the previous event schedule.
- 13753 b. If the previous event is executing, the End Device shall change directly from its current state to the  
13754 requested state at the *Effective Start Time* of the *Overlapping Event* (Note: Rule #4 effects *Effective Start*  
13755 *Time*). The End Device returns a *Report Event Status* command (referencing the previous event) with the  
13756 Event Status set to 0x07 (the event has been superseded).
- 13757 6. Randomization **shall not** cause event conflicts or unmanaged gaps. To clarify:
- 13758 a. When event starting randomization is requested, time periods between the *Start Time* of an event and the  
13759 *Effective Start Time* a device should either maintain its current state or apply changes which contribute  
13760 to energy saving. Preference would be to maintain current state.
- 13761 b. When event ending randomization is used and the *Effective End Time* overlaps the *Effective Start Time*  
13762 of a *Successive Event*, the *Effective Start Time* takes precedence. Events are not reported as superseded,  
13763 End devices should report event status as it would a normal set of *Successive Events*.
- 13764 c. It is recommended devices apply the same Start and Stop Randomization values for consecutive events to  
13765 help prevent unexpected gaps between events.
- 13766 d. Devices **shall not** artificially create a gap between *Successive Events*.
- 13767 7. It is permissible to have gaps when events are not *Successive Events* or *Overlapping Events*.
- 13768 8. If multiple device classes are identified for an event, future events for individual device classes (or a subset  
13769 of the original event) that cause an *Overlapping Event* will supersede the original event strictly for that  
13770 device class (or a subset of the original event). Note: Rule #5 applies to all *Overlapping Events*.

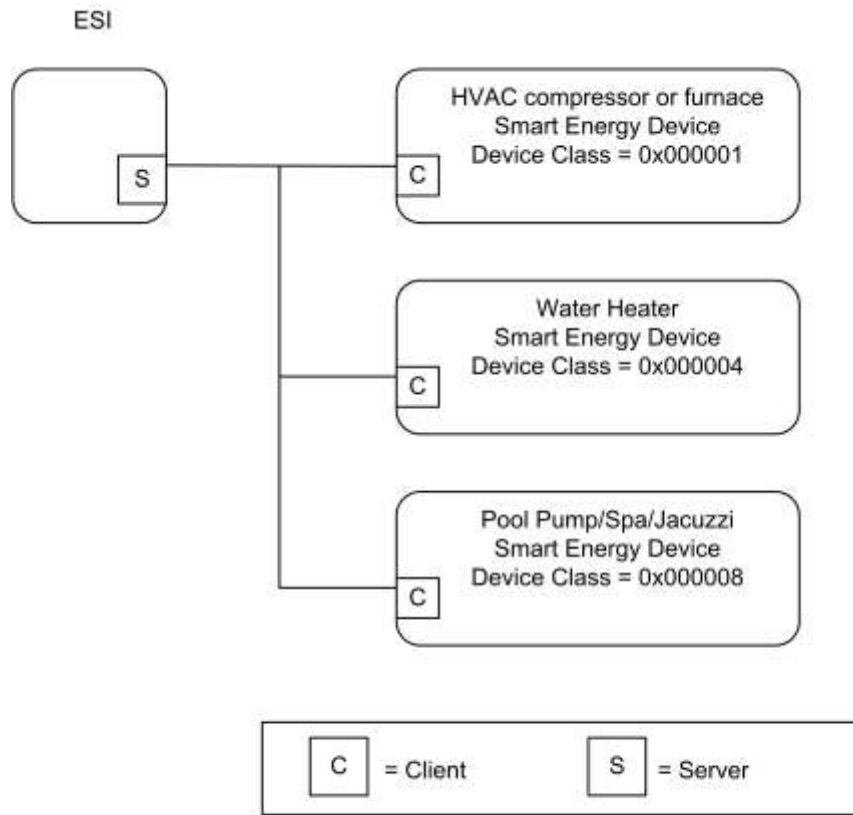
### 13771 10.3.5.3 Event Examples

13772 Smart Energy devices which act upon Demand Response and Load Control events shall use the following examples  
13773 for understanding and managing overlapping and superseded events. Within those examples, references to  
13774 multiple device classes will be used. Figure 10-20 depicts a representation of those devices in a Smart Energy  
13775 network.



13776

Figure 10-20. Smart Energy Device Class Reference Example



Note: Device names are examples for illustration purposes only

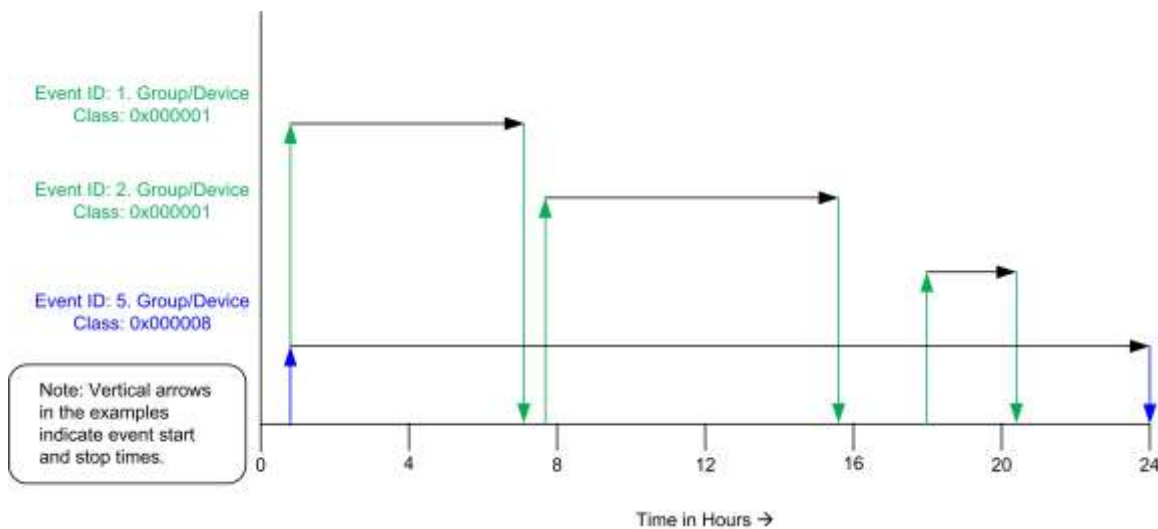
13777

13778 **10.3.5.3.1 Correct Overlapping Events for Different Device Classes**

13779 Figure 10-21 depicts a correct series of DR/LC event for device class of 0x000001 (reference for the BitMap  
 13780 definition) with an event scheduled for another device class during the same period.

13781

Figure 10-21. Correctly Overlapping Events



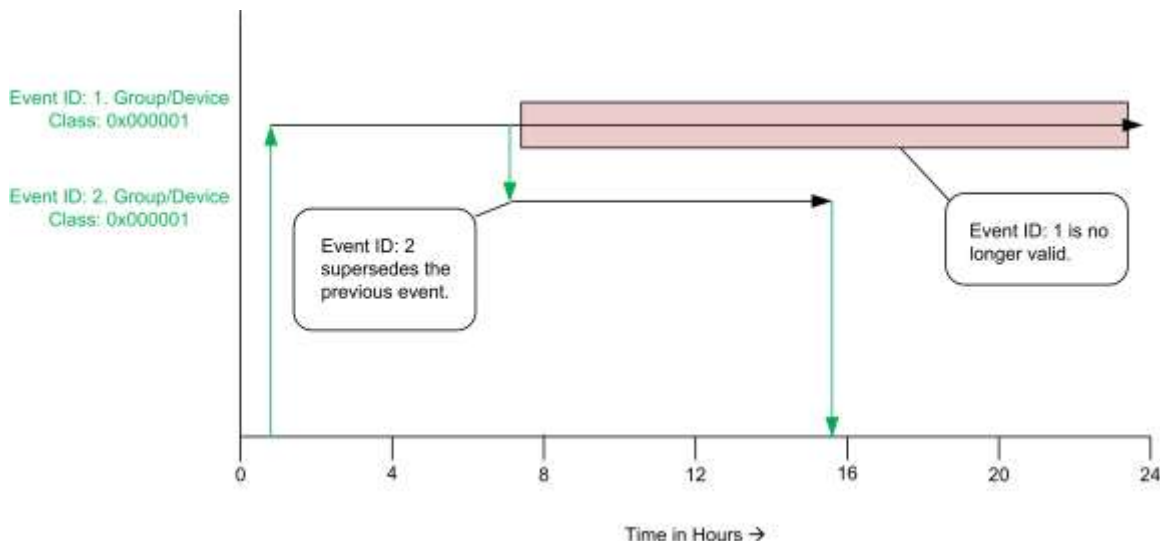
13782  
 13783

13784 In Figure 10-21, Device Class 0x000001 receives a sequence of 3 unique DR/LC events to be scheduled and  
 13785 acted upon. During this same 24 hour period, Device Class 0x000008 receives one scheduled DR/LC event that  
 13786 spans across the same time period as the events scheduled for Device Class 0x0000001. Because both Device  
 13787 Classes are unique, there are no conflicts due to Overlapping Events.

13788 **10.3.5.3.2 Correct Superseded Event for a Device Class**

13789 Figure 10-22 depicts a correct series of DR/LC events for device class of 0x000001 (reference for the BitMap  
 13790 definition) where an event is scheduled then later superseded.

13791 **Figure 10-22. Correct Superseding of Events**



13792  
 13793  
 13794 In Figure 10-22, Device Class 0x000001 receives DR/LC Event ID#1 setup for a 24 hour Scheduled Period, which  
 13795 later is superseded by DR/LC Event ID#2, invalidating the remainder of Event ID#1, which is cancelled.

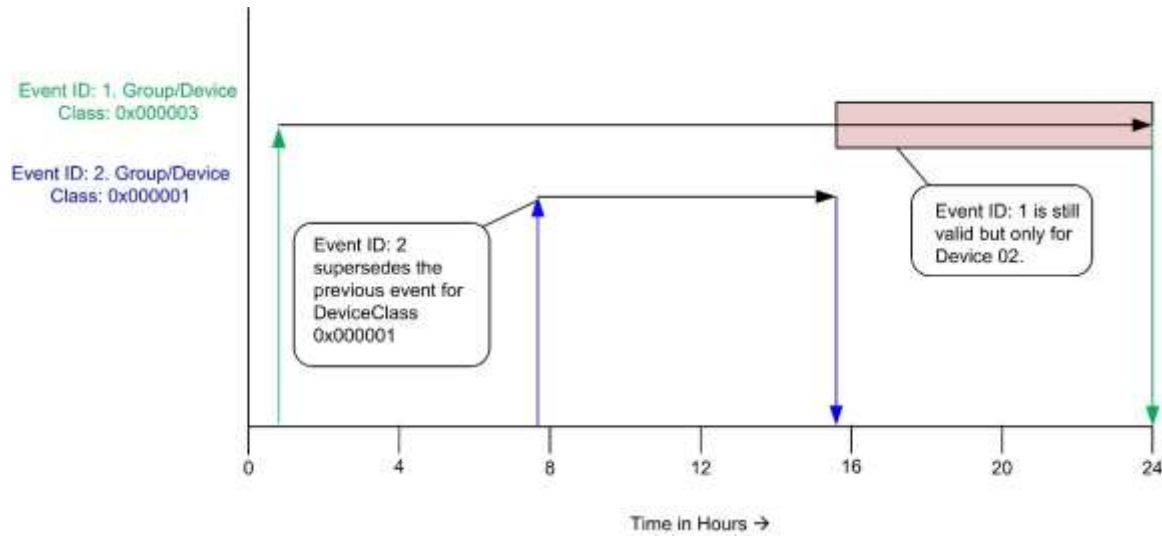
13796 **10.3.5.3.3 Superseding Events for Subsets of Device Classes**

13797 Figure 10-23 depicts a correct series of DR/LC events for device class of 0x000001 (reference for the BitMap  
 13798 definition) with an event scheduled for another device class during the same time period.

13799

Figure 10-23. Superseded Event for a Subset of Device Classes

13800



13801

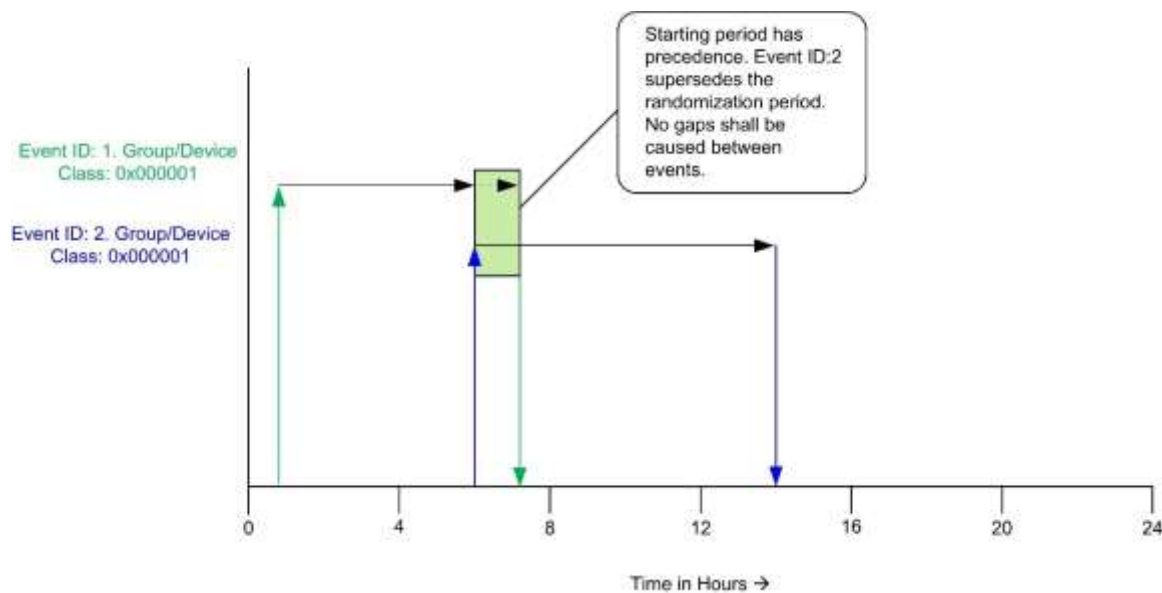
13802 In Figure 10-23, Device Class 0x000003 receives DR/LC Event ID#1 setup for a 24 hour Scheduled Period, which  
 13803 is targeted for both Device Class 0x000002 and 0x000001 (OR'ed == 0x000003). In the example, Event ID#2 is  
 13804 issued only for Device Class 0x000001, invalidating the remainder of Event ID#1 for that device class. DR/LC  
 13805 Event ID#1 is still valid for Device Class 0x000002, which in the example should run to completion.

13806 **10.3.5.3.4 Ending Randomization Between Events**

13807 Figure 10-24 depicts an Effective End Time that overlaps a second scheduled DR/LC event for device class of  
 13808 0x000001 (reference for the BitMap definition).

13809

Figure 10-24. Ending Randomization Between Events



13810

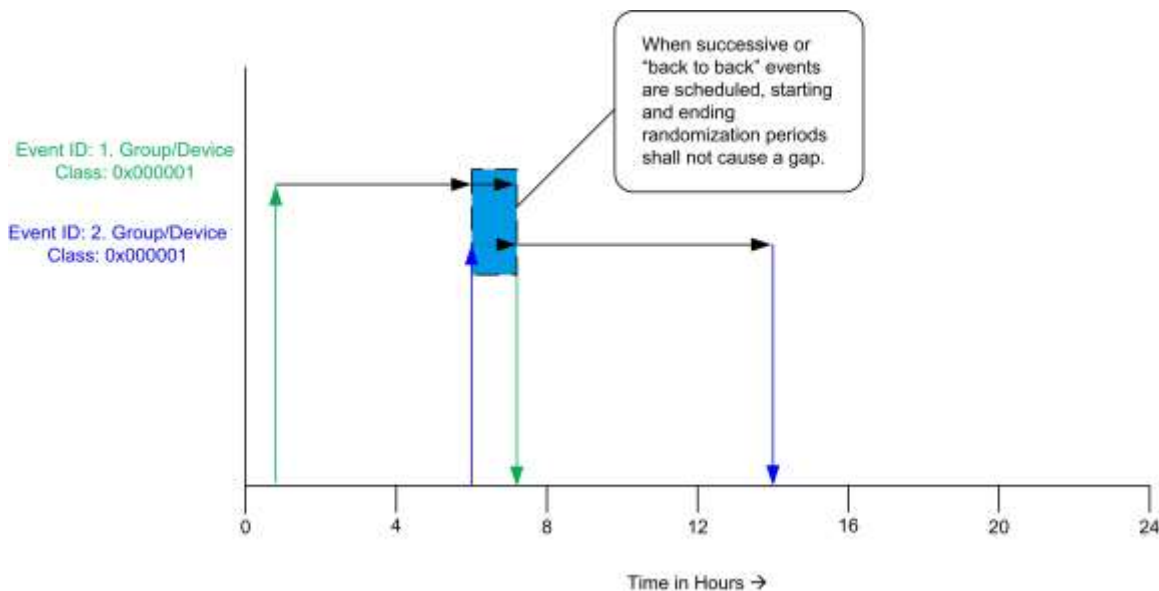
13811 In Figure 10-24, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting  
 13812 (please refer to sub-clause 10.3.2.3.1.1.1 for more detail). A second DR/LC (Event ID#2) is issued with a

13813 starting time which matches the ending time of DR/LC Event ID#1. In this situation, the Start Time of Event ID#2  
13814 has precedence. Event ID#1 is not reported as superseded.

13815 **10.3.5.3.5 Start Randomization Between Events**

13816 Figure 10-25 depicts an Effective Start Time that overlaps a previously scheduled DR/LC event for device class of  
13817 0x000001 (reference for the BitMap definition).

13818 **Figure 10-25. Start Randomization Between Events**



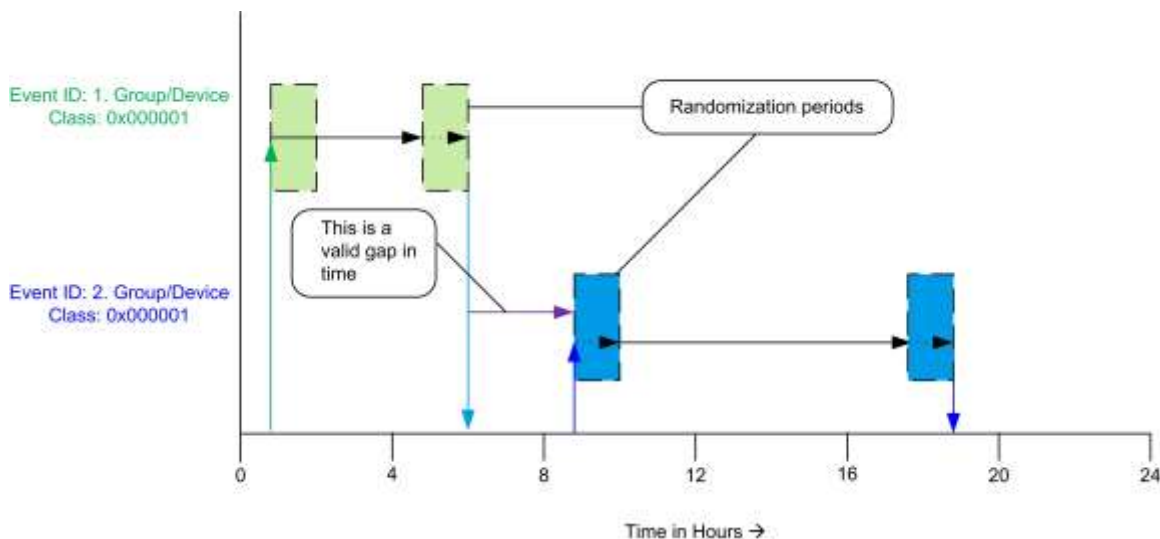
13819 In Figure 10-25, Device Class 0x000001 receives a DR/LC Event ID#1 with an ending randomization setting  
13820 (please refer to sub-clause 10.3.2.3.1.1.1 for more detail). Effective End Time of Event ID#1 is not known. A  
13821 second DR/LC (Event ID#2) is issued with a starting randomized setting, which has an Effective Start Time that  
13822 could overlap or start after the Effective End Time of DR/LC Event ID#1. In this situation, the Effective Start  
13823 Time of Event ID#2 has precedence but the DR/LC device must also prevent any artificial gaps caused by the  
13824 Effective Start Time of Event ID#2 and Effective End Time of Event ID#1.  
13825  
13826

13827 **10.3.5.3.6 Acceptable Gaps Caused by Start and Stop Randomization of Events**

13828 Figure 10-26 depicts an acceptable gap between two scheduled DR/LC events for device class of 0x000001  
13829 (reference for the BitMap definition) using both starting and ending randomization with both events.

13830

**Figure 10-26. Acceptable Gaps with Start and Stop Randomization**



13831

13832

13833 In Figure 10-26, Device Class 0x000001 receives a DR/LC Event ID#1 with both a starting and ending  
 13834 randomization setting. (Please refer to sub-clause 10.3.2.3.1.1.1 for more detail). A second DR/LC Event ID#2 is  
 13835 also issued with both a starting and ending randomized setting. The primary configuration to note in this example is  
 13836 the Effective End Time of DR/LC Event ID#1 completes well in advance of the Effective Start Time of DR/LC  
 13837 Event ID#2. In this scenario, regardless of randomization a gap is naturally created by the scheduling of the  
 13838 events and is acceptable.

## 13839 10.4 Metering

### 13840 10.4.1 Overview

13841 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 13842 etc.

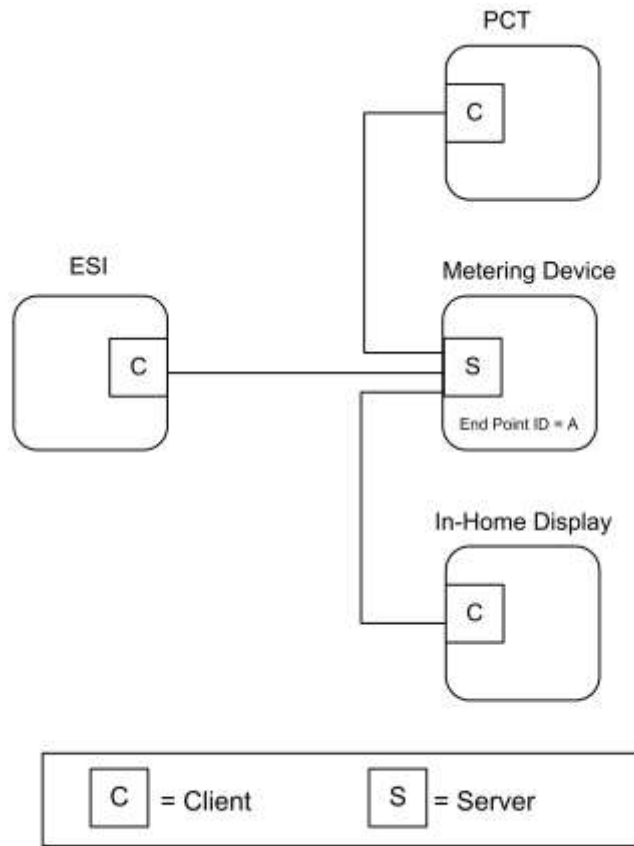
13843 The Metering Cluster provides a mechanism to retrieve usage information from Electric, Gas, Water, and  
 13844 potentially Thermal metering devices. These devices can operate on either battery or mains power, and can have a  
 13845 wide variety of sophistication. The Metering Cluster is designed to provide flexibility while limiting capabilities to a  
 13846 set number of metered information types. More advanced forms or data sets from metering devices will be  
 13847 supported in the Smart Energy Tunneling Cluster, which will be defined in sub-clause 10.6.

13848 The following figures identify three configurations as examples utilizing the Metering Cluster.

13849 In Figure 10-27, the metering device is the source of information provided via the Metering Cluster Server.

13850

Figure 10-27. Standalone ESI Model with Mains Powered Metering Device



13851

*Note: Device names are examples for illustration purposes only*

13852

13853

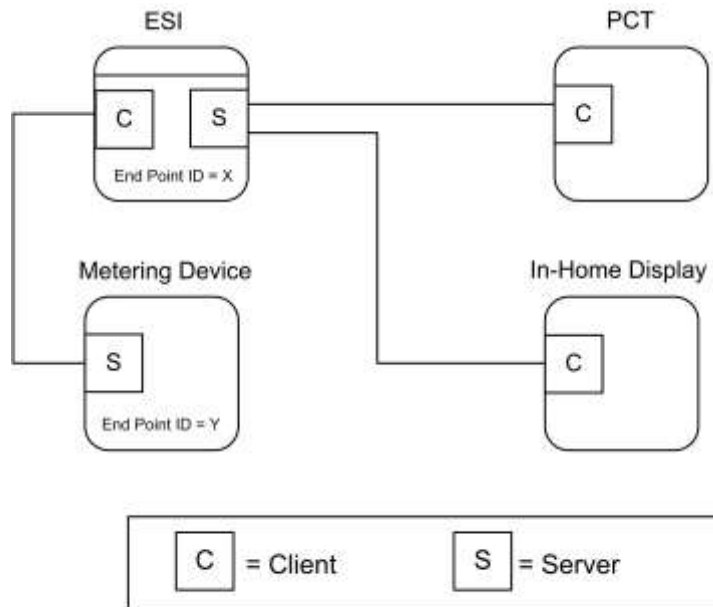
13854

13855

In the example shown in Figure 10-28, the metering device is running on battery power and its duty cycle for providing information is unknown. It's expected the ESI will act like a mirrored image or a mailbox (Client) for the metering device data, allowing other Smart Energy devices to gain access to the metering device's data (provided via an image of its Metering Cluster).

13856

**Figure 10-28. Standalone ESI Model with Battery Powered Metering Device**



13857

*Note: Device names are examples for illustration purposes only*

13858

13859

13860

13861

13862

13863

13864

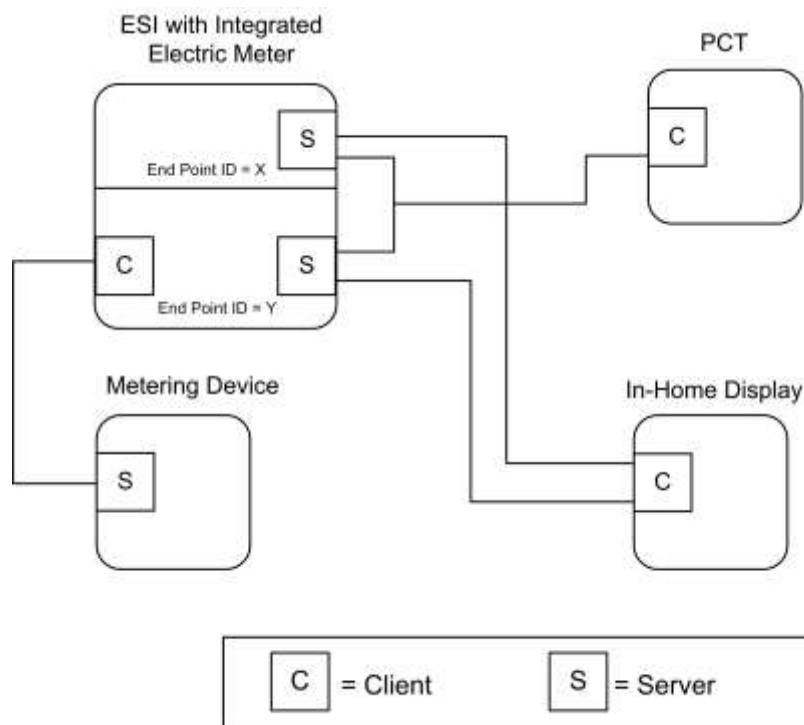
13865

13866

In the example shown in Figure 10-29, much like the previous example in Figure 10-28, the external metering device is running on battery power and its duty cycle for providing information is unknown. It's expected the ESI will act like a Client side mailbox for the external metering device data, allowing other Smart Energy devices to gain access to the metering device's data (provided via an image of its Metering Cluster). Since the ESI can also contain an integrated metering device where its information is also conveyed through the Metering Cluster, each device (external metering device mailbox and integrated meter) will be available via independent EndPoint IDs. Other Smart Energy devices that need to access the information must understand the ESI cluster support by performing service discoveries. It can also identify if an Endpoint ID is a mailbox/ mirror of a metering device by reading the MeteringDeviceType attribute (refer to sub-clause 10.4.2.2.4.7).

13867

**Figure 10-29. ESI Model with Integrated Metering Device**



*Note: Device names are examples for illustration purposes only*

13868

13869 In the above examples (Figure 10-28 and Figure 10-29), it's expected the ESI would perform Attribute Reads  
 13870 (or configure Attribute Reporting) and use the GetProfile command to receive the latest information whenever  
 13871 the Metering Device (EndPoint Z) wakes up. When received, the ESI will update its mailbox (EndPoint ID Y in  
 13872 Figure 10-28 and Figure 10-29) to reflect the latest data available. A metering device using the mirror is also  
 13873 allowed (and recommended) to push metering data updates to the ESI via Report Attribute commands as  
 13874 described in sub-clause 10.4.3.4.

13875 Other Smart Energy devices can access EndPoint Y in the ESI to receive the latest information just as they would  
 13876 to access information in the ESI's integrated Electric meter (as in Figure 10-29, EndPoint X) and other Metering  
 13877 devices (as in Figure 10-27, EndPoint A).

13878 **10.4.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

13879 **10.4.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEMT	Type 1 (client to server)



13880 **10.4.1.3 Cluster Identifiers**

Identifier	Name
0x0702	Metering (Smart Energy)

13881 **10.4.2 Server**

13882 **10.4.2.1 Dependencies**

13883 Subscribed reporting of Metering attributes.

13884 **10.4.2.2 Attributes**

13885 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 13886 contain up to 256 attributes. Attribute identifiers are encoded such that the most significant octet specifies the  
 13887 attribute set and the least significant octet specifies the attribute within the set. The currently defined attribute sets  
 13888 are listed in Table 10-29.

13889 **Note:** Certain attributes within this cluster are provisionary and not certifiable. Refer to the individual attribute sets  
 13890 for details of the relevant attributes.

13891 **Table 10-29. Metering Cluster Attribute Sets**

Attribute Set Identifier	Description
0x00	Reading Information Set
0x01	TOU Information Set
0x02	Meter Status
0x03	Formatting
0x04	Historical Consumption
0x05	Load Profile Configuration
0x06	Supply Limit
0x07	Block Information
0x08	Alarms

13892 **10.4.2.2.1 Reading Information Set**

13893 The set of attributes shown in Table 10-30 provides a remote access to the reading of the Electric, Gas, or Water  
 13894 metering device. A reading must support at least one register which is the actual total summation of the  
 13895 delivered quantity (kWh, m<sup>3</sup>, ft<sup>3</sup>, ccf, US gl).

13896 Please note: In the following attributes, the term “Delivered” refers to the quantity of Energy, Gas, or Water that  
 13897 was delivered to the customer from the utility. Likewise, the term “Received” refers to the quantity of Energy,  
 13898 Gas, or Water that was received by the utility from the customer.

13899 **Note:** Metering Cluster Reading Attributes 0x10-0x14 in this revision of this specification are provisionary  
 13900 and not certifiable. This feature set may change before reaching certifiable status in a future revision of this  
 13901 specification.

13902

Table 10-30. Reading Information Attribute Set

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>CurrentSummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	M
0x0001	<i>CurrentSummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0002	<i>CurrentMaxDemandDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0003	<i>CurrentMaxDemandReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0004	<i>DFTSummation</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0005	<i>DailyFreezeTime</i>	uint16	0x0000 to 0x183C	R	0x0000	O
0x0006	<i>PowerFactor</i>	int8	-100 to +100	R	0x00	O
0x0007	<i>ReadingSnapShotTime</i>	UTC		R	-	O
0x0008	<i>CurrentMaxDemandDeliveredTime</i>	UTC		R	-	O
0x0009	<i>CurrentMaxDemandReceivedTime</i>	UTC		R	-	O
0x000A	<i>DefaultUpdatePeriod</i>	uint8	0x00 to 0xFF	R	0x1E	O
0x000B	<i>FastPollUpdatePeriod</i>	uint8	0x00 to 0xFF	R	0x05	O
0x000C	<i>CurrentBlockPeriodConsumptionDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x000D	<i>DailyConsumptionTarget</i>	uint24	0x000000 to 0xFFFFFF	R	-	O
0x000E	<i>CurrentBlock</i>	enum8	0x00 to 0x10	R	-	O
0x000F	<i>ProfileIntervalPeriod</i>	enum8	0x00 to 0xFF	R	-	O
0x0010	<i>IntervalReadReportingPeriod</i>	uint16	0x0000 to 0xFFFF	R	0	O
0x0011	<i>PresetReadingTime</i>	uint16	0x0000 to 0x173B	R	0x0000	O
0x0012	<i>VolumePerReport</i>	uint16	0x0000 to 0xFFFF	R	-	O
0x0013	<i>FlowRestriction</i>	uint8	0x00 to 0xFF	R	-	O

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0014	<i>Supply Status</i>	enum8	0x00 to 0xFF	R	-	O
0x0015	<i>CurrentInletEnergyCarrierSummation</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	M*
0x0016	<i>CurrentOutletEnergyCarrierSummation</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0017	<i>InletTemperature</i>	int24	-8,388,607 to -----	R	-	M*
0x0018	<i>OutletTemperature</i>	int24	-8,388,607 to 8,388,607	R	-	M*
0x0019	<i>ControlTemperature</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x001A	<i>CurrentInletEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x001B	<i>CurrentOutletEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x001C	<i>PreviousBlockPeriodConsumptionDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

13903

13904 \* *Mandatory for Heat or Cooling; Optional for others*

13905 **10.4.2.2.1.1 CurrentSummationDelivered Attribute**

13906 CurrentSummationDelivered represents the most recent summed value of Energy, Gas, or Water delivered and  
 13907 consumed in the premises. CurrentSummationDelivered is mandatory and must be provided as part of the minimum  
 13908 data set to be provided by the metering device. CurrentSummationDelivered is updated continuously as new  
 13909 measurements are made.

13910 **10.4.2.2.1.2 CurrentSummationReceived Attribute**

13911 CurrentSummationReceived represents the most recent summed value of Energy, Gas, or Water generated and  
 13912 delivered from the premises. If optionally provided, CurrentSummationReceived is updated continuously as new  
 13913 measurements are made.

13914 **10.4.2.2.1.3 CurrentMaxDemandDelivered Attribute**

13915 CurrentMaxDemandDelivered represents the maximum demand or rate of delivered value of Energy, Gas, or Water  
 13916 being utilized at the premises. If optionally provided, CurrentMaxDemandDelivered is updated continuously as  
 13917 new measurements are made.

13918 **10.4.2.2.1.4 CurrentMaxDemandReceived Attribute**

13919 CurrentMaxDemandReceived represents the maximum demand or rate of received value of Energy, Gas, or Water  
 13920 being utilized by the utility. If optionally provided, CurrentMaxDemandReceived is updated continuously as new  
 13921 measurements are made.

- 13922 **10.4.2.2.1.5** *DFTSummation Attribute*
- 13923 DFTSummation represents a snapshot of attribute CurrentSummationDelivered captured at the time indicated by  
13924 attribute DailyFreezeTime. If optionally provided, DFTSummation is updated once every 24 hours and captured at  
13925 the time set in sub-clause 10.4.2.2.1.6.
- 13926 **10.4.2.2.1.6** *DailyFreezeTime Attribute*
- 13927 DailyFreezeTime represents the time of day when DFTSummation is captured. DailyFreezeTime is an unsigned 16-  
13928 bit value representing the hour and minutes for DFT. The byte usages are:
- 13929 **Bits 0 to 7:** Range of 0 to 0x3C representing the number of minutes past the top of the hour.
- 13930 **Bits 8 to 15:** Range of 0 to 0x17 representing the hour of the day (in 24-hour format).
- 13931 **10.4.2.2.1.7** *PowerFactor Attribute*
- 13932 PowerFactor contains the Average Power Factor ratio in 1/100ths. Valid values are 0 to 99.
- 13933 **10.4.2.2.1.8** *ReadingSnapShotTime Attribute*
- 13934 The ReadingSnapShotTime attribute represents the last time all of the CurrentSummationDelivered,  
13935 CurrentSummationReceived, CurrentMaxDemandDelivered, and CurrentMaxDemandReceived attributes that are  
13936 supported by the device were updated.
- 13937 **10.4.2.2.1.9** *CurrentMaxDemandDeliveredTime Attribute*
- 13938 The CurrentMaxDemandDeliveredTime attribute represents the time when CurrentMaxDemandDelivered reading  
13939 was captured.
- 13940 **10.4.2.2.1.10** *CurrentMaxDemandReceivedTime Attribute*
- 13941 The CurrentMaxDemandReceivedTime attribute represents the time when CurrentMaxDemandReceived reading  
13942 was captured.
- 13943 **10.4.2.2.1.11** *DefaultUpdatePeriod Attribute*
- 13944 The DefaultUpdatePeriod attribute represents the interval (seconds) at which the InstantaneousDemand attribute is  
13945 updated when not in fast poll mode. InstantaneousDemand may be continuously updated as new measurements  
13946 are acquired, but at a minimum InstantaneousDemand must be updated at the DefaultUpdatePeriod. The  
13947 DefaultUpdatePeriod may apply to other attributes as defined by the device manufacturer.
- 13948 **10.4.2.2.1.12** *FastPollUpdatePeriod Attribute*
- 13949 The FastPollUpdatePeriod attribute represents the interval (seconds) at which the InstantaneousDemand attribute is  
13950 updated when in fast poll mode. InstantaneousDemand may be continuously updated as new measurements are  
13951 acquired, but at a minimum, InstantaneousDemand must be updated at the FastPollUpdatePeriod. The  
13952 FastPollUpdatePeriod may apply to other attributes as defined by the device manufacturer.
- 13953 **10.4.2.2.1.13** *CurrentBlockPeriodConsumptionDelivered Attribute*
- 13954 The CurrentBlockPeriodConsumptionDelivered attribute represents the most recent summed value of Energy, Gas  
13955 or Water delivered and consumed in the premises during the Block Tariff Period.
- 13956 The CurrentBlockPeriodConsumptionDelivered is reset at the start of each Block Tariff Period.
- 13957 **10.4.2.2.1.14** *DailyConsumptionTarget Attribute*
- 13958 The DailyConsumptionTarget attribute is a daily target consumption amount that can be displayed to the consumer  
13959 on a HAN device, with the intent that it can be used to compare to actual daily consumption (e.g. compare to the  
13960 CurrentDayConsumptionDelivered).
- 13961 This may be sent from the utility to the ESI, or it may be derived. Although intended to be based on Block  
13962 Thresholds, it can be used for other targets not related to blocks. The formatting will be based on the  
13963 HistoricalConsumptionFormatting attribute.

13964 Example: If based on a Block Threshold, the DailyConsumptionTarget could be calculated based on the number of  
 13965 days specified in the Block Tariff Period and a given Block Threshold as follows:  $\text{DailyConsumptionTarget} =$   
 13966  $\text{BlockNThreshold} / ((\text{BlockPeriodDuration} / 60) / 24)$ . Example: If the target is based on a Block1Threshold of  
 13967 675kWh and where 43200 BlockThresholdPeriod is the number of minutes in the billing period (30 days), the  
 13968 ConsumptionDailyTarget would be  $675 / ((43200 / 60) / 24) = 22.5$  kWh per day.

13969 **10.4.2.2.1.15 CurrentBlock Attribute**

13970 When Block Tariffs are enabled, CurrentBlock is an 8-bit Enumeration which indicates the currently active  
 13971 block. If blocks are active then the current active block is based on the CurrentBlockPeriodConsumptionDelivered  
 13972 and the block thresholds. Block 1 is active when the value of CurrentBlockPeriodConsumptionDelivered is less  
 13973 than Block1Threshold value; Block 2 is active when CurrentBlockPeriodConsumptionDelivered is greater than  
 13974 Block1Threshold value and less than Block2Threshold value, and so on. Block 16 is active when the value of  
 13975 CurrentBlockPeriodConsumptionDelivered is greater than Block15Threshold value.

13976 **Table 10-31. Block Enumerations**

Enumerated Value	Register Block
0x00	No Blocks in use
0x01	Block1
0x02	Block2
0x03	Block3
0x04	Block4
0x05	Block5
0x06	Block6
0x07	Block7
0x08	Block8
0x09	Block9
0x0A	Block10
0x0B	Block11
0x0C	Block12
0x0D	Block13
0x0E	Block14
0x0F	Block15
0x10	Block16

13977 **10.4.2.2.1.16 ProfileIntervalPeriod Attribute**

13978 The ProfileIntervalPeriod attribute is currently included in the Get Profile Response command payload, but does not  
 13979 appear in an attribute set. This represents the duration of each interval. ProfileIntervalPeriod represents the  
 13980 interval or time frame used to capture metered Energy, Gas, and Water consumption for profiling purposes. The  
 13981 enumeration for this field shall match one of the ProfileIntervalPeriod values defined in sub-clause 10.4.2.3.1.1.1.

13982 **10.4.2.2.1.17 IntervalReadReportingPeriod Attribute**

13983 The IntervalReadReportingPeriod attribute represents how often (in minutes) the water or gas meter is to wake up  
 13984 and provide interval data. E.g.: If IntervalReadReportingPeriod is set to 360, then every 6 hours the water or gas

13985 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  
13986 every 4 days it will wake up and provide 4 days of interval data in a Get Profile Response command. In some  
13987 cases data may overlap data sent in previous Get Profile Response command.

#### 13988 **10.4.2.2.1.18 PresetReadingTime Attribute**

13989 The PresetReadingTime attribute represents the time of day (in quarter hour increments) at which the meter  
13990 will wake up and report a register reading even if there has been no consumption for the previous 24 hours.  
13991 PresetReadingTime is an unsigned 16-bit value representing the hour and minutes. The byte usages are:

13992 **Bits 0 to 7:** Range of 0 to 0x3B representing the number of minutes past the top of the hour.

13993 **Bits 8 to 15:** Range of 0 to 0x17 representing the hour of the day (in 24-hour format).

13994 E.g.: A setting of 0x172D would represent 23:45 hours or 11:45 pm; a setting of 0x071E would represent 07:30  
13995 hours or 7:30 am. A setting of 0xFFFF indicates this feature is disabled. The use of Attribute Reporting  
13996 Configuration is optional.

#### 13997 **10.4.2.2.1.19 VolumePerReport Attribute**

13998 The VolumePerReport attribute represents the volume per report increment from the water or gas meter. For  
13999 example a gas meter might be set to report its register reading for every time 1 cubic meter of gas is used. For a  
14000 water meter it might report the register value every 10 liters of water usage.

#### 14001 **10.4.2.2.1.20 FlowRestriction Attribute**

14002 The FlowRestriction attribute represents the volume per minute limit set in the flow restrictor. This applies to  
14003 water but not for gas. A setting of 0xFF indicates this feature is disabled.

#### 14004 **10.4.2.2.1.21 SupplyStatus Attribute**

14005 The SupplyStatus attribute represents the state of the supply at the customer's premises. The enumerated values  
14006 for this field are outlined in Table 10-32.

14007 **Table 10-32. Supply Status Attribute Enumerations**

Enumerated Value	Status
0x00	Supply OFF
0x01	Supply OFF/ARMED
0x02	Supply ON

#### 14008 **10.4.2.2.1.22 CurrentInletEnergyCarrierSummation Attribute**

14009 CurrentInletEnergyCarrierSummation is the current integrated volume of a given energy carrier measured on the  
14010 inlet. The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and  
14011 EnergyCarrierSummationFormatting attributes (refer to Table 10-40).

14012 The Energy consumption registered in CurrentSummationDelivered is not necessarily a direct function of this  
14013 value. The quality of the energy carrier may vary from day to day, e.g. Gas may have different quality.

14014 For heat and cooling meters the energy carrier is water at high or low temperature, the energy withdrawn from such  
14015 a system is a function of the flow and the inlet and outlet temperature.

#### 14016 **10.4.2.2.1.23 CurrentOutletEnergyCarrierSummation Attribute**

14017 CurrentOutletEnergyCarrierSummation is the current integrated volume of a given energy carrier measured on the  
14018 outlet. The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and  
14019 EnergyCarrierSummationFormatting attributes (refer to Table 10-40).

#### 14020 **10.4.2.2.1.24 InletTemperature Attribute**

- 14021 InletTemperature is the temperature measured on the energy carrier inlet.
- 14022 The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and  
 14023 TemperatureFormatting attributes (refer to Table 10-40).
- 14024 **10.4.2.2.1.25 OutletTemperature Attribute**
- 14025 OutletTemperature is the temperature measured on the energy carrier outlet.
- 14026 The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and  
 14027 TemperatureFormatting attributes (refer to Table 10-40).
- 14028 **10.4.2.2.1.26 ControlTemperature Attribute**
- 14029 ControlTemperature is a reference temperature measured on the meter used to validate the Inlet/Outlet  
 14030 temperatures.
- 14031 The formatting and unit of measure for this value is specified in the TemperatureUnitOfMeasure and  
 14032 TemperatureFormatting attributes (refer to Table 10-40).
- 14033 **10.4.2.2.1.27 CurrentInletEnergyCarrierDemand Attribute**
- 14034 CurrentInletEnergyCarrierDemand is the current absolute demand on the energy carrier inlet.
- 14035 The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and  
 14036 EnergyCarrierDemandFormatting attributes (refer to Table 10-40).
- 14037 For a heat or cooling meter this will be the current absolute flow rate measured on the inlet.
- 14038 **10.4.2.2.1.28 CurrentOutletEnergyCarrierDemand Attribute**
- 14039 CurrentOutletEnergyCarrierDemand is the current absolute demand on the energy carrier outlet.
- 14040 The formatting and unit of measure for this value is specified in the EnergyCarrierUnitOfMeasure and  
 14041 EnergyCarrierDemandFormatting attributes (refer to Table 10-40).
- 14042 For a heat or cooling meter this will be the current absolute flow rate measured on the outlet.
- 14043 **10.4.2.2.1.29 PreviousBlockPeriodConsumptionDelivered Attribute**
- 14044 The PreviousBlockPeriodConsumptionDelivered attribute represents the total value of Energy, Gas or Water  
 14045 delivered and consumed in the premises at the end of the previous Block Tariff Period. If supported, the  
 14046 PreviousBlockPeriodConsumptionDelivered attribute is updated at the end of each Block Tariff Period.
- 14047 **10.4.2.2.2 Summation TOU Information Set**
- 14048 The set of attributes shown in Table 10-33 provides a remote access to the Electric, Gas, or Water metering  
 14049 device’s Time of Use (TOU) readings.
- 14050 **Note:** TOU Information Attribute Set Attributes 0x0C-0x1D in this revision of this specification are provisional  
 14051 and not certifiable. This feature set may change before reaching certifiable status in a future revision of this  
 14052 specification.

14053 **Table 10-33. TOU Information Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0100	<i>CurrentTier1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0101	<i>CurrentTier1SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0102	<i>CurrentTier2SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0103	<i>CurrentTier2SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0104	<i>CurrentTier3SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0105	<i>CurrentTier3SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0106	<i>CurrentTier4SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0107	<i>CurrentTier4SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0108	<i>CurrentTier5SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0109	<i>CurrentTier5SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010A	<i>CurrentTier6SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010B	<i>CurrentTier6SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010C	<i>CurrentTier7SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010D	<i>CurrentTier7SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010E	<i>CurrentTier8SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x010F	<i>CurrentTier8SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0110	<i>CurrentTier9SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O



<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0111	<i>CurrentTier9SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0112	<i>CurrentTier10SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0113	<i>CurrentTier10SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0114	<i>CurrentTier11SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0115	<i>CurrentTier11SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0116	<i>CurrentTier12SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0117	<i>CurrentTier12SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0118	<i>CurrentTier13SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0119	<i>CurrentTier13SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x011A	<i>CurrentTier14SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x011B	<i>CurrentTier1SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x011C	<i>CurrentTier15SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x011D	<i>CurrentTier15SummationReceived</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

14054 **10.4.2.2.2.1** *CurrentTierNSummationDelivered* Attributes

14055 Attributes *CurrentTier1SummationDelivered* through *CurrentTierNSummationDelivered* represent the most recent  
 14056 summed value of Energy, Gas, or Water delivered to the premises (i.e. delivered to the customer from the utility)  
 14057 at a specific price tier as defined by a TOU schedule or a real time pricing period. If optionally provided, attributes  
 14058 *CurrentTier1SummationDelivered* through *CurrentTierNSummationDelivered* are updated continuously as new  
 14059 measurements are made.

14060 **10.4.2.2.2.2** *CurrentTierNSummationReceived* Attributes

14061 Attributes *CurrentTier1SummationReceived* through *CurrentTierNSummationReceived* represent the most recent  
 14062 summed value of Energy, Gas, or Water provided by the premises (i.e. received by the utility from the customer)  
 14063 at a specific price tier as defined by a TOU schedule or a real time pricing period. If optionally provided, attributes  
 14064 *CurrentTier1SummationReceived* through *CurrentTierNSummationReceived* are updated continuously as new  
 14065 measurements are made.

#### 14066 10.4.2.2.3 Meter Status Attribute Set

14067 The Meter Status Attribute Set is defined in Table 10-34.

14068 **Table 10-34. Meter Status Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0200	<i>Status</i>	map8	0x00 to 0xFF	R	0x00	M
0x0201	<i>RemainingBatteryLife</i>	uint8	0x00 to 0xFF	R	-	O
0x0202	<i>HoursInOperation</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	M:Heat M:Cooling O:others
0x0203	<i>HoursInFault</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O

#### 14069 10.4.2.2.3.1 Status Attribute

14070 The Status attribute provides indicators reflecting the current error conditions found by the metering device.  
 14071 This attribute is an 8-bit field where when an individual bit is set, an error or warning condition exists. The  
 14072 behavior causing the setting or resetting each bit is device specific. In other words, the application within the  
 14073 metering device will determine and control when these settings are either set or cleared. Depending on the  
 14074 commodity type, the bits of this attribute will take on different meaning. Table 10-35 through Table 10-38 show the  
 14075 bit mappings for the Status attribute for Electricity, Gas, Water and Heating/Cooling, respectively. A battery-  
 14076 operated meter will report any change in state of the Status when it wakes up via a ZCL report attributes  
 14077 command. The ESI is expected to make alarms available to upstream systems together with consumption data  
 14078 collected from the battery operated meter.

14079 **Table 10-35. Mapping of the Status Attribute (Electricity)**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Service Disconnect Open	Leak Detect	Power Quality	Power Failure	Tamper Detect	Low Battery	Check Meter

14080  
 14081 The definitions of the Electricity Status bits are:

14082 **Service Disconnect Open:** Set to true when the service have been disconnected to this premises.

14083 **Leak Detect:** Set to true when a leak have been detected.

14084 **Power Quality:** Set to true if a power quality event have been detected such as a low voltage, high voltage.

14085 **Power Failure:** Set to true during a power outage.

14086 **Tamper Detect:** Set to true if a tamper event has been detected.

14087 **Low Battery:** Set to true when the battery needs maintenance.

14088 **Check Meter:** Set to true when a non fatal problem has been detected on the meter such as a measurement  
 14089 error, memory error, and self check error.

14090 **Table 10-36. Meter Status Attribute (Gas)**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reverse Flow	Service Disconnect	Leak Detect	Low Pressure	Not Defined	Tamper Detect	Low Battery	Check Meter

14091

14092 The definitions of the Gas Status bits are:

14093 **Reverse Flow:** Set to true if flow detected in the opposite direction to normal (from consumer to supplier).

14094 **Service Disconnect:** Set to true when the service has been disconnected to this premises. Ex. The valve is in the  
 14095 closed position preventing delivery of gas.

14096 **Leak Detect:** Set to true when a leak has been detected.

14097 **Low Pressure:** Set to true when the pressure at the meter is below the meter's low pressure threshold value.

14098 **Tamper Detect:** Set to true if a tamper event has been detected.

14099 **Low Battery:** Set to true when the battery needs maintenance.

14100 **Check Meter:** Set to true when a non fatal problem has been detected on the meter such as a measurement  
 14101 error, memory error, or self check error.

14102 **Table 10-37. Meter Status Attribute (Water)**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reverse Flow	Service Disconnect	Leak Detect	Low Pressure	Pipe Empty	Tamper Detect	Low Battery	Check Meter

14103

14104 The definitions of the Water Status bits are:

14105 **Reverse Flow:** Set to true if flow detected in the opposite direction to normal (from consumer to supplier).

14106 **Service Disconnect:** Set to true when the service has been disconnected to this premises. Ex. The valve is in the  
 14107 closed position preventing delivery of water.

14108 **Leak Detect:** Set to true when a leak has been detected.

14109 **Low Pressure:** Set to true when the pressure at the meter is below the meter's low pressure threshold value.

14110 **Pipe Empty:** Set to true when the service pipe at the meter is empty and there is no flow in either direction.

14111 **Tamper Detect:** Set to true if a tamper event has been detected.

14112 **Low Battery:** Set to true when the battery needs maintenance.

14113 **Check Meter:** Set to true when a non fatal problem has been detected on the meter such as a measurement  
 14114 error, memory error, or self check error.

14115 **Table 10-38. Meter Status Attribute (Heat and Cooling)**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-------	-------	-------	-------	-------	-------	-------	-------

Flow Sensor	Service Disconnect	Leak Detect	Burst Detect	Temperature Sensor	Tamper Detect	Low Battery	Check Meter
-------------	--------------------	-------------	--------------	--------------------	---------------	-------------	-------------

14116

14117 The definitions of the Heat and Cooling Status bits are:

14118 **Flow Sensor:** Set to true when an error is detected on a flow sensor at this premises.14119 **Service Disconnect:** Set to true when the service has been disconnected to this premises. Ex. The valve is in the  
14120 closed position preventing delivery of heat or cooling.14121 **Leak Detect:** Set to true when a leak has been detected.14122 **Burst Detect:** Set to true when a burst is detected on pipes at this premises.14123 **Temperature Sensor:** Set to true when an error is detected on a temperature sensor at this premises.14124 **Tamper Detect:** Set to true if a tamper event has been detected.14125 **Low Battery:** Set to true when the battery needs maintenance.14126 **Check Meter:** Set to true when a non fatal problem has been detected on the meter such as a measurement  
14127 error, memory error, or self check error.14128 **Note:** It is not necessary to set aside Bit 7 as an “Extension Bit” for future expansion. If extra status bits are  
14129 required an Extended Meter Status attribute may be added to support additional status values.14130 **10.4.2.2.3.2 RemainingBatteryLife Attribute**14131 RemainingBatteryLife represents the estimated remaining life of the battery in % of capacity. A setting of 0xFF  
14132 indicates this feature is disabled. The range 0 - 100 where 100 = 100%, 0xFF = Unknown.14133 **10.4.2.2.3.3 HoursInOperation Attribute**14134 HoursInOperation is a counter that increments once every hour during operation. This may be used as a check for  
14135 tampering.14136 **Note:** For meters that are not electricity meters turning off the meter does not necessarily prevent delivery of  
14137 energy—but the meter might not be able to measure it.14138 **10.4.2.2.3.4 HoursInFault Attribute**14139 HoursInFault is a counter that increments once every hour when the device is in operation with a fault detected.  
14140 This may be used as a check for tampering.14141 **Note:** For meters that are not electricity meters turning off the meter does not necessarily prevent delivery of  
14142 energy—but the meter might not be able to measure it.14143 **10.4.2.2.4 Formatting**14144 The following set of attributes provides the ratios and formatting hints required to transform the received  
14145 summations, consumptions, temperatures, or demands/ rates into displayable values. If the Multiplier and  
14146 Divisor attribute values are non-zero, they are used in conjunction with the SummationFormatting,  
14147 ConsumptionFormatting, DemandFormatting, and TemperatureFormatting attributes.

14148 Equations required to accomplish this task are defined below:

14149  $\text{Summation} = \text{Summation received} * \text{Multiplier} / \text{Divisor}$   
14150 (formatted using SummationFormatting)14151  $\text{Consumption} = \text{Consumption received} * \text{Multiplier} / \text{Divisor}$   
14152 (formatted using ConsumptionFormatting)14153  $\text{Demand} = \text{Demand received} * \text{Multiplier} / \text{Divisor}$   
14154 (formatted using DemandFormatting)

- 14155 Temperature = Temperature received \* Multiplier / Divisor
- 14156 If the Multiplier and Divisor attribute values are zero, just the formatting hints defined in SummationFormatting,  
 14157 ConsumptionFormatting, DemandFormatting and TemperatureFormatting attributes are used.
- 14158 The summation received, consumption received, demand received, and temperature received variables used above  
 14159 can be replaced by any of the attributes listed in sub-clauses 10.4.2.2.4.4, 10.4.2.2.4.5, 10.4.2.2.4.6,  
 14160 10.4.2.2.4.11, 10.4.2.2.4.12, and 10.4.2.2.4.14.
- 14161 Table 10-39 shows examples that demonstrate the relation between these attributes.

**Table 10-39. Formatting Examples**

Attribute	Example 1	Example 2	Example 3
Value as transmitted and received	52003	617	23629
Unit of Measure	kWh	CCF	kWh
Multiplier	1	2	6
Divisor	1000	100	10000
Number of Digits to the left of the Decimal Point	5	4	5
Number of Digits to the right of the Decimal Point	0	2	3
Suppress leading zeros	False	False	True
Displayed value	00052	0012.34	14.177

- 14163
- 14164 The Consumption Formatting Attribute Set is defined in Table 10-40.
- 14165 **Note:** Consumption Formatting Attribute 0x07 in this revision of this specification is provisional and not  
 14166 certifiable. This feature set may change before reaching certifiable status in a future revision of this specification.

**Table 10-40. Formatting Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0300	<i>UnitofMeasure</i>	enum8	0x00 to 0xFF	R	0x00	M
0x0301	<i>Multiplier</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0302	<i>Divisor</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0303	<i>SummationFormatting</i>	map8	0x00 to 0xFF	R	-	M
0x0304	<i>DemandFormatting</i>	map8	0x00 to 0xFF	R	-	O
0x0305	<i>HistoricalConsumptionFormatting</i>	map8	0x00 to 0xFF	R	-	O
0x0306	<i>MeteringDeviceType</i>	map8	0x00 to 0xFF	R	-	M
0x0307	<i>SiteID</i>	octstr	1 to 33 octets	R	-	O

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0308	<i>MeterSerialNumber</i>	octstr	1 to 25 octets	R	-	O
0x0309	<i>EnergyCarrierUnitOfMeasure</i>	enum8	0x00 to 0xFF	R	-	M*
0x030A	<i>EnergyCarrierSummationFormatting</i>	map8	0x00 to 0xFF	R	-	M*
0x030B	<i>EnergyCarrierDemandFormatting</i>	map8	0x00 to 0xFF	R	-	O
0x030C	<i>TemperatureUnitOfMeasure</i>	enum8	0x00 to 0xFF	R	-	M*
0x030D	<i>TemperatureFormatting</i>	map8	0x00 to 0xFF	R	-	M*

14168

14169 \* *Mandatory for Heat or Cooling; Optional for others*14170 **10.4.2.2.4.1 UnitofMeasure Attribute**

14171 UnitofMeasure provides a label for the Energy, Gas, or Water being measured by the metering device. The units of  
 14172 measure apply to all summations, consumptions/ profile interval and demand/rate supported by this cluster. Other  
 14173 measurements such as the power factor are self describing. This attribute is an 8-bit enumerated field. The bit  
 14174 descriptions for this Attribute are listed in Table 10-41.

14175

**Table 10-41. UnitofMeasure Attribute Enumerations**

<b>Values</b>	<b>Description</b>
0x00	kWh (Kilowatt Hours) & kW (Kilowatts) in pure binary format
0x01	m <sup>3</sup> (Cubic Meter) & m <sup>3</sup> /h (Cubic Meter per Hour) in pure binary format
0x02	ft <sup>3</sup> (Cubic Feet) & ft <sup>3</sup> /h (Cubic Feet per Hour) in pure binary format
0x03	ccf ((100 or Centum) Cubic Feet) & ccf/h ((100 or Centum) Cubic Feet per Hour) in pure binary format
0x04	US gl (US Gallons) & US gl/h (US Gallons per Hour) in pure binary format.
0x05	IMP gl (Imperial Gallons) & IMP gl/h (Imperial Gallons per Hour) in pure binary format
0x06	BTUs & BTU/h in pure binary format
0x07	Liters & l/h (Liters per Hour) in pure binary format
0x08	kPA (gauge) in pure binary format
0x09	kPA (absolute) in pure binary format
0x0A	mcf (1000 Cubic Feet) & mcf/h (1000 Cubic feet per hour) in pure binary format
0x0B	Unitless in pure binary format
0x0C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in pure binary format
0x80	kWh (Kilowatt Hours) & kW (Kilowatts) in BCD format
0x81	m <sup>3</sup> (Cubic Meter) & m <sup>3</sup> /h (Cubic Meter per Hour) in BCD format
0x82	ft <sup>3</sup> (Cubic Feet) & ft <sup>3</sup> /h (Cubic Feet per Hour) in BCD format

Values	Description
0x83	ccf ((100 or Centum) Cubic Feet) & ccf/h ((100 or Centum) Cubic Feet per Hour) in BCD format
0x84	US gl (US Gallons) & US gl/h (US Gallons per Hour) in BCD format
0x85	IMP gl (Imperial Gallons) & IMP gl/h (Imperial Gallons per Hour) in BCD format
0x86	BTUs & BTU/h in BCD format
0x87	Liters & l/h (Liters per Hour) in BCD format
0x88	kPA (gauge) in BCD format
0x89	kPA (absolute) in BCD format
0x8A	mcf (1000 Cubic Feet) & mcf/h (1000 Cubic Feet per Hour) in BCD format
0x8B	unitless in BCD format
0x8C	MJ (Mega Joule) and MJ/s (Mega Joule per second (MW)) in BCD format

14176

14177 **Note:** When using BCD for meter reads, the values A to F are special values or indicators denoting “Opens”,  
 14178 “Shorts”, and etc. conditions when reading meter register hardware. Any SE device displaying the BCD based  
 14179 values to end users should use a non-decimal value to replace the A to F. In other words, a device could use an  
 14180 “\*” in place of the special values or indicators.

14181 **10.4.2.2.4.2 Multiplier Attribute**

14182 Multiplier provides a value to be multiplied against a raw or uncompensated sensor count of Energy, Gas, or Water  
 14183 being measured by the metering device. If present, this attribute must be applied against all summation,  
 14184 consumption and demand values to derive the delivered and received values expressed in the unit of measure  
 14185 specified. This attribute must be used in conjunction with the Divisor attribute.

14186 **10.4.2.2.4.3 Divisor Attribute**

14187 Divisor provides a value to divide the results of applying the Multiplier Attribute against a raw or uncompensated  
 14188 sensor count of Energy, Gas, or Water being measured by the metering device. If present, this attribute must be  
 14189 applied against all summation, consumption and demand values to derive the delivered and received values  
 14190 expressed in the unit of measure specified. This attribute must be used in conjunction with the Multiplier attribute.

14191 **10.4.2.2.4.4 SummationFormatting Attribute**

14192 SummationFormatting provides a method to properly decipher the number of digits and the decimal location of  
 14193 the values found in the Summation Information Set of attributes. This attribute is to be decoded as follows:

14194 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14195 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14196 **Bit 7:** If set, suppress leading zeros.

14197 This attribute shall be used against the following attributes:

- 14198 • CurrentSummationDelivered
- 14199 • CurrentSummationReceived
- 14200 • TOU Information attributes
- 14201 • DFTSummation
- 14202 • Block Information attributes

**14203 10.4.2.2.4.5 DemandFormatting Attribute**

14204 DemandFormatting provides a method to properly decipher the number of digits and the decimal location of the  
14205 values found in the Demand-related attributes. This attribute is to be decoded as follows:

14206 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14207 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14208 **Bit 7:** If set, suppress leading zeros.

14209 This attribute shall be used against the following attributes:

- 14210 • CurrentMaxDemandDelivered
- 14211 • CurrentMaxDemandReceived
- 14212 • InstantaneousDemand

**14213 10.4.2.2.4.6 HistoricalConsumptionFormatting Attribute**

14214 HistoricalConsumptionFormatting provides a method to properly decipher the number of digits and the decimal  
14215 location of the values found in the Historical Consumption Set of attributes. This attribute is to be decoded as  
14216 follows:

14217 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14218 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14219 **Bit 7:** If set, suppress leading zeros.

14220 This attribute shall be used against the following attributes:

- 14221 • CurrentDayConsumptionDelivered
- 14222 • CurrentDayConsumptionReceived
- 14223 • PreviousDayConsumptionDelivered
- 14224 • PreviousDayConsumptionReceived
- 14225 • CurrentPartialProfileIntervalValue
- 14226 • Intervals
- 14227 • DailyConsumptionTarget

**14228 10.4.2.2.4.7 MeteringDeviceType Attribute**

14229 MeteringDeviceType provides a label for identifying the type of metering device present. The attribute are values  
14230 representing Energy, Gas, Water, Thermal, Heat, Cooling, and mirrored metering devices. The defined values are  
14231 represented in Table 10-42. (Note that these values represent an Enumeration, and not an 8-bit bitmap as  
14232 indicated in the attribute description. For backwards compatibility reasons, the data type has not been changed,  
14233 though the data itself should be treated like an enum.)

14234 Where a mirror is provided for a battery-powered metering device, the mirror shall assume the relevant  
14235 'Mirrored Metering' device type (128-133) whilst the meter itself shall utilize the 'Metering' device type (1 to 6).  
14236 It shall be the responsibility of the device providing the mirror to modify the Device Type shown on the mirror  
14237 to that of a 'Mirrored Metering' device.

14238 **Table 10-42. MeteringDeviceType Attribute**

Values	Description
0	Electric Metering
1	Gas Metering



Values	Description
2	Water Metering
3	Thermal Metering (deprecated)
4	Pressure Metering
5	Heat Metering
6	Cooling Metering
128	Mirrored Gas Metering
129	Mirrored Water Metering
130	Mirrored Thermal Metering (deprecated)
131	Mirrored Pressure Metering
132	Mirrored Heat Metering
133	Mirrored Cooling Metering

14239

14240 **Note:** Heat and cooling meters are used for measurement and billing of heat (and cooling) delivered through liquid  
 14241 (water) based central heating systems. The consumers are typically billed by the kWh, calculated from the  
 14242 flow and the temperatures in and out.

14243 **10.4.2.2.4.8 SiteID Attribute**

14244 The SiteID is a ZCL Octet String field capable of storing a 32 character string (the first octet indicates length)  
 14245 encoded in UTF-8 format. The SiteID is a text string, known in the UK as the M-PAN number for electricity,  
 14246 MPRN for gas and 'Stand Point' in South Africa. These numbers specify the meter point location in a  
 14247 standardized way. The field is defined to accommodate the number of characters typically found in the UK and  
 14248 Europe (16 digits). Generally speaking the field is numeric but is defined for the possibility of an alpha-numeric  
 14249 format by specifying an octet string.

14250 **10.4.2.2.4.9 MeterSerialNumber Attribute**

14251 The MeterSerialNumber is a ZCL Octet String field capable of storing a 24 character string (the first octet  
 14252 indicates length) encoded in UTF-8 format. It is used to provide a unique identification of the metering device.

14253 **10.4.2.2.4.10 EnergyCarrierUnitOfMeasure Attribute**

14254 The EnergyCarrierUnitOfMeasure specifies the unit of measure that the EnergyCarrier is measured in. This unit of  
 14255 measure is typically a unit of volume or flow and cannot be an amount of energy. The enumeration of this  
 14256 attribute is otherwise identical to the UnitofMeasure attribute (Table 10-41).

14257 **10.4.2.2.4.11 EnergyCarrierSummationFormatting Attribute**

14258 EnergyCarrierSummationFormatting provides a method to properly decipher the number of digits and the decimal  
 14259 location of the values found in the Summation- related attributes.

14260 This attribute is to be decoded as follows:

14261 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14262 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14263 **Bit 7:** If set, suppress leading zeros.

14264 This attribute shall be used in relation with the following attributes:

- 14265 • CurrentInletEnergyCarrierSummation

- 14266
- CurrentOutletEnergyCarrierSummation

14267 **10.4.2.2.4.12 EnergyCarrierDemandFormatting Attribute**

14268 EnergyCarrierDemandFormatting provides a method to properly decipher the number of digits and the decimal  
14269 location of the values found in the Demand-related attributes.

14270 This attribute is to be decoded as follows:

14271 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14272 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14273 **Bit 7:** If set, suppress leading zeros.

14274 This attribute shall be used in relation with the following attributes:

- 14275
- CurrentInletEnergyCarrierDemand
  - 14276 • CurrentOutletEnergyCarrierDemand
  - 14277 • CurrentDayMaxEnergyCarrierDemand
  - 14278 • PreviousDayMaxEnergyCarrierDemand
  - 14279 • CurrentMonthMaxEnergyCarrierDemand
  - 14280 • CurrentMonthMinEnergyCarrierDemand
  - 14281 • CurrentYearMinEnergyCarrierDemand
  - 14282 • CurrentYearMaxEnergyCarrierDemand

14283 **10.4.2.2.4.13 TemperatureUnitOfMeasure Attribute**

14284 The TemperatureUnitOfMeasure specifies the unit of measure that temperatures are measured in. The enumeration  
14285 of this attribute is shown in Table 10-43.

14286 **Table 10-43. TemperatureUnitOfMeasure Enumeration**

Values	Description
0x00	K (Degrees Kelvin) in pure Binary format.
0x01	°C (Degrees Celsius) in pure Binary format.
0x02	°F (Degrees Fahrenheit) in pure Binary format.
0x80	K (Degrees Kelvin) in BCD format.
0x81	°C (Degrees Celsius) in BCD format.
0x82	°F (Degrees Fahrenheit) in BCD format.

14287 **10.4.2.2.4.14 TemperatureFormatting Attribute**

14288 TemperatureFormatting provides a method to properly decipher the number of digits and the decimal location of  
14289 the values found in the Temperature-related attributes. This attribute is to be decoded as follows:

14290 **Bits 0 to 2:** Number of Digits to the right of the Decimal Point.

14291 **Bits 3 to 6:** Number of Digits to the left of the Decimal Point.

14292 **Bit 7:** If set, suppress leading zeros.

14293 This attribute shall be used in relation with the following attributes:

- 14294
- InletTemperature

- 14295 • OutletTemperature
- 14296 • ControlTemperature

14297 **10.4.2.2.5 Historical Consumption Attribute**

14298 The Historical Attribute Set is defined in Table 10-44.

14299 **Note:** Historical Consumption Attributes 0x09-0x0E, 0x11 and 0x12 in this revision of this specification are  
 14300 provisional and not certifiable. This feature set may change before reaching certifiable status in a future revision of  
 14301 this specification.

14302 **Table 10-44. Historical Consumption Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0400	<i>InstantaneousDemand</i>	int24	-8,388,607 to 8,388,607	R	0	O
0x0401	<i>CurrentDayConsumptionDelivered</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0402	<i>CurrentDayConsumptionReceived</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0403	<i>PreviousDayConsumptionDelivered</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0404	<i>PreviousDayConsumptionReceived</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0405	<i>CurrentPartialProfileIntervalStartTimeDelivered</i>	UTC		R	-	O
0x0406	<i>CurrentPartialProfileIntervalStartTimeReceived</i>	UTC	0x000000 to 0xFFFFFFFF	R	-	O
0x0407	<i>CurrentPartialProfileIntervalValueDelivered</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0408	<i>CurrentPartialProfileIntervalValueReceived</i>	uint24	0x000000 to 0xFFFFFFFF	R	-	O
0x0409	<i>CurrentDayMaxPressure</i>	uint48	0x000000 000000 to 0xFFFFFFFF FFFFFFFF	R	-	O
0x040a	<i>CurrentDayMinPressure</i>	uint48	0x000000 000000 to 0xFFFFFFFF FFFFFFFF	R	-	O
0x040b	<i>PreviousDayMaxPressure</i>	uint48	0x000000 000000 to 0xFFFFFFFF FFFFFFFF	R	-	O
0x040c	<i>PreviousDayMinPressure</i>	uint48	0x000000 000000 to 0xFFFFFFFF FFFFFFFF	R	-	O
0x040d	<i>CurrentDayMaxDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x040e	<i>PreviousDayMaxDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x040f	<i>CurrentMonthMaxDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0410	<i>CurrentYearMaxDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0411	<i>CurrentDayMaxEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0412	<i>PreviousDayMaxEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0413	<i>CurrentMonthMaxEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0414	<i>CurrentMonthMinEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0415	<i>CurrentYearMaxEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O
0x0416	<i>CurrentYearMinEnergyCarrierDemand</i>	int24	-8,388,607 to 8,388,607	R	-	O

- 14303 **10.4.2.2.5.1 *InstantaneousDemand Attribute***  
14304 InstantaneousDemand represents the current Demand of Energy, Gas, or Water delivered or received at the  
14305 premises. Positive values indicate demand delivered to the premises where negative values indicate demand  
14306 received from the premises. InstantaneousDemand is updated continuously as new measurements are made. The  
14307 frequency of updates to this field is specific to the metering device, but should be within the range of once every  
14308 second to once every 5 seconds.
- 14309 **10.4.2.2.5.2 *CurrentDayConsumptionDelivered Attribute***  
14310 CurrentDayConsumptionDelivered represents the summed value of Energy, Gas, or Water generated and delivered  
14311 to the premises since midnight local time. If optionally provided, CurrentDayConsumptionDelivered is updated  
14312 continuously as new measurements are made.
- 14313 **10.4.2.2.5.3 *CurrentDayConsumptionReceived Attribute***  
14314 CurrentDayConsumptionReceived represents the summed value of Energy, Gas, or Water generated and received  
14315 from the premises since midnight local time. If optionally provided, CurrentDayConsumptionReceived is updated  
14316 continuously as new measurements are made.
- 14317 **10.4.2.2.5.4 *PreviousDayConsumptionDelivered Attribute***  
14318 PreviousDayConsumptionDelivered represents the summed value of Energy, Gas, or Water generated and delivered  
14319 to the premises within the previous 24 hour period starting at midnight local time. If optionally provided,  
14320 CurrentDayConsumptionDelivered is updated every midnight local time.
- 14321 **10.4.2.2.5.5 *PreviousDayConsumptionReceived Attribute***  
14322 PreviousDayConsumptionReceived represents the summed value of Energy, Gas, or Water generated and received  
14323 from the premises within the previous 24 hour period starting at midnight local time. If optionally provided,  
14324 CurrentDayConsumptionReceived is updated every midnight local time.
- 14325 **10.4.2.2.5.6 *CurrentPartialProfileIntervalStartTimeDelivered Attribute***  
14326 CurrentPartialProfileIntervalStartTimeDelivered represents the start time of the current Load Profile interval being  
14327 accumulated for commodity delivered.
- 14328 **10.4.2.2.5.7 *CurrentPartialProfileIntervalStartTimeReceived Attribute***  
14329 CurrentPartialProfileIntervalStartTimeReceived represents the start time of the current Load Profile interval being  
14330 accumulated for commodity received.
- 14331 **10.4.2.2.5.8 *CurrentPartialProfileIntervalValueDelivered Attribute***  
14332 CurrentPartialProfileIntervalValueDelivered represents the value of the current Load Profile interval being  
14333 accumulated for commodity delivered.
- 14334 **10.4.2.2.5.9 *CurrentPartialProfileIntervalValueReceived Attribute***  
14335 CurrentPartialProfileIntervalValueReceived represents the value of the current Load Profile interval being  
14336 accumulated for commodity received.
- 14337 **10.4.2.2.5.10 *CurrentDayMaxPressure Attribute***  
14338 CurrentDayMaxPressure is the maximum pressure reported during a day from the water or gas meter.
- 14339 **10.4.2.2.5.11 *PreviousDayMaxPressure Attribute***  
14340 PreviousDayMaxPressure represents the maximum pressure reported during previous day from the water or gas  
14341 meter.
- 14342 **10.4.2.2.5.12 *CurrentDayMinPressure Attribute***  
14343 CurrentDayMinPressure is the minimum pressure reported during a day from the water or gas meter.

- 14344 **10.4.2.2.5.13**     ***PreviousDayMinPressure Attribute***  
14345 PreviousDayMinPressure represents the minimum pressure reported during previous day from the water or gas  
14346 meter.
- 14347 **10.4.2.2.5.14**     ***CurrentDayMaxDemand Attribute***  
14348 CurrentDayMaxDemand represents the maximum demand or rate of delivered value of Energy, Gas, or Water  
14349 being utilized at the premises.
- 14350 **10.4.2.2.5.15**     ***PreviousDayMaxDemand Attribute***  
14351 PreviousDayMaxDemand represents the maximum demand or rate of delivered value of Energy, Gas, or Water  
14352 being utilized at the premises.
- 14353 **Note:** At the end of a day the metering device will transfer the CurrentDayMaxPressure into  
14354 PreviousDayMaxPressure, CurrentDayMinPressure into PreviousDayMinPressure and CurrentDayMaxDemand into  
14355 PreviousDayMaxDemand.
- 14356 **10.4.2.2.5.16**     ***CurrentMonthMaxDemand Attribute***  
14357 CurrentMonthMaxDemand is the maximum demand reported during a month from the meter.  
14358 For electricity, heat and cooling meters this is the maximum power reported in a month.
- 14359 **10.4.2.2.5.17**     ***CurrentYearMaxDemand Attribute***  
14360 CurrentYearMaxDemand is the maximum demand reported during a year from the meter.  
14361 For electricity, heat and cooling meters this is the maximum power reported in a year.
- 14362 **10.4.2.2.5.18**     ***CurrentDayMaxEnergyCarrierDemand Attribute***  
14363 CurrentDayMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a day from the  
14364 meter.  
14365 **Note:** At the end of a day the meter will transfer the CurrentDayMaxEnergyCarrierDemand into  
14366 PreviousDayMaxEnergyCarrierDemand.  
14367 For heat and cooling meters this is the maximum flow rate on the inlet reported in a day.
- 14368 **10.4.2.2.5.19**     ***PreviousDayMaxEnergyCarrierDemand Attribute***  
14369 PreviousDayMaxEnergyCarrierDemand is the maximum energy carrier demand reported during the previous day  
14370 from the meter.
- 14371 **10.4.2.2.5.20**     ***CurrentMonthMaxEnergyCarrierDemand Attribute***  
14372 CurrentMonthMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a month from the  
14373 meter.  
14374 For heat and cooling meters this is the maximum flow rate on the inlet reported in a month.
- 14375 **10.4.2.2.5.21**     ***CurrentMonthMinEnergyCarrierDemand Attribute***  
14376 CurrentMonthMinEnergyCarrierDemand is the minimum energy carrier demand reported during a month from the  
14377 meter.  
14378 For heat and cooling meters this is the minimum flow rate on the inlet reported in a month.  
14379 **Note:** This attribute may be used to detect leaks if there has been no flow rate of zero in the last month.
- 14380 **10.4.2.2.5.22**     ***CurrentYearMaxEnergyCarrierDemand Attribute***  
14381 CurrentYearMaxEnergyCarrierDemand is the maximum energy carrier demand reported during a year from the  
14382 meter.

14383 For heat and cooling meters this is the maximum flow rate on the inlet reported in a year.

#### 14384 **10.4.2.2.5.23** *CurrentYearMinEnergyCarrierDemand Attribute*

14385 CurrentYearMinEnergyCarrierDemand is the minimum energy carrier demand reported during a year from the  
14386 heat meter.

14387 For heat and cooling meters this is the minimum flow rate on the inlet reported in a year.

14388 **Note:** This attribute may be used to detect leaks if there has been no flow rate of zero in the last year

### 14389 **10.4.2.2.6** **Load Profile Configuration**

14390 The Load Profile Configuration Attribute Set is defined in Table 10-45.

14391 **Table 10-45. Load Profile Configuration Attribute Set**

Identifier	Name	Type	Range	Acc	Default	M/O
0x0500	<i>MaxNumberOfPeriodsDelivered</i>	uint8	0x00 to 0xFF	R	0x18	O

#### 14392 **10.4.2.2.6.1** *MaxNumberOfPeriodsDelivered Attribute*

14393 MaxNumberOfPeriodsDelivered represents the maximum number of intervals the device is capable of returning in  
14394 one Get Profile Response command. It is required MaxNumberOfPeriodsDelivered fit within the default  
14395 Fragmentation ASDU size of 128 bytes, or an optionally agreed upon larger Fragmentation ASDU size  
14396 supported by both devices. Please refer to [Z1] for further details on Fragmentation settings.

### 14397 **10.4.2.2.7** **Supply Limit Attributes**

14398 This set of attributes is used to implement a “Supply Capacity Limit” program where the demand at the premises  
14399 is limited to a preset consumption level over a preset period of time. Should this preset limit be exceeded the  
14400 meter could interrupt supply to the premises or to devices within the premises. The supply limit information in  
14401 this attribute set can be used by In-Home displays, PCTs, or other devices to display a warning when the  
14402 supply limit is being approached. The Supply Limit Attribute Set is defined in Table 10-46.

14403 **Table 10-46. Supply Limit Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0600	<i>CurrentDemandDelivered</i>	uint24	0x000000 to 0xFFFFFFFF	R		O
0x0601	<i>DemandLimit</i>	uint24	0x000000 to 0xFFFFFFFF	R		O
0x0602	<i>DemandIntegrationPeriod</i>	uint8	0x01 to 0xFF	R	-	O
0x0603	<i>NumberOfDemandSubintervals</i>	uint8	0x01 to 0xFF	R	-	O

#### 14404 **10.4.2.2.7.1** *CurrentDemandDelivered Attribute*

14405 CurrentDemandDelivered represents the current Demand of Energy, Gas, or Water delivered at the premises.

14406 CurrentDemandDelivered may be continuously updated as new measurements are acquired, but at a minimum

14407 CurrentDemandDelivered must be updated at the end of each integration sub- period, which can be obtained by

14408 dividing the DemandIntegrationPeriod by the NumberOfDemandSubintervals.

14409 This attribute shall be adjusted using the Multiplier and Divisor attributes found in the Formatting Attribute Set and  
 14410 can be formatted using the DemandFormatting attribute. The final result represents an engineering value in the  
 14411 unit defined by the UnitofMeasure attribute.

14412 **10.4.2.2.7.2 DemandLimit Attribute**

14413 DemandLimit reflects the current supply demand limit set in the meter. This value can be compared to the  
 14414 CurrentDemandDelivered attribute to understand if limits are being approached or exceeded.

14415 Adjustment and formatting of this attribute follow the same rules as the CurrentDemandDelivered.

14416 A value of “0xFFFFFFFF” indicates “demand limiting” is switched off.

14417 **10.4.2.2.7.3 DemandIntegrationPeriod Attribute**

14418 DemandIntegrationPeriod is the number of minutes over which the CurrentDemandDelivered attribute is  
 14419 calculated. Valid range is 0x01 to 0xFF. 0x00 is a reserved value.

14420 **10.4.2.2.7.4 NumberOfDemandSubintervals Attribute**

14421 NumberOfDemandSubintervals represents the number of subintervals used within the DemandIntegrationPeriod.  
 14422 The subinterval duration (in minutes) is obtained by dividing the DemandIntegrationPeriod by the  
 14423 NumberOfDemandSubintervals. The CurrentDemandDelivered attribute is updated at each subinterval. Valid range  
 14424 is 0x01 to 0xFF. 0x00 is a reserved value.

14425 As a Rolling Demand example, DemandIntegrationPeriod could be set at 30 (for 30 minute period) and  
 14426 NumberOfDemandSubintervals could be set for 6. This would provide 5 minute (30/6 = 5) subinterval periods.

14427 As a Block Demand example, DemandIntegrationPeriod could be set at 30 (for 30 minute period) and  
 14428 NumberOfDemandSubintervals could be set for 1. This would provide a single 30 minute subinterval period.

14429 **10.4.2.2.8 Block Information Set**

14430 The set of attributes shown in Table 10-47 provides a remote access to the Electric, Gas, or Water metering  
 14431 device's block readings. The Block Information attribute set supports Block pricing and combined Tier-Block  
 14432 pricing, the number of blocks is one greater than the number of block thresholds defined in the Pricing cluster.

14433 **Table 10-47. Block Information Attribute Set**

Id	Name	Type	Range	Acc	Def	M/O
0x0700	<i>CurrentNoTierBlock1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0701	<i>CurrentNoTierBlock2SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0702	<i>CurrentNoTierBlock3SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x070N	<i>CurrentNoTierBlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x070f	<i>CurrentNoTierBlock16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0710	<i>CurrentTier1Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x0711	<i>CurrentTier1Block2SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0712	<i>CurrentTier1Block3SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x071N	<i>CurrentTier1BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x071f	<i>CurrentTier1Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0720	<i>CurrentTier2Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x072N	<i>CurrentTier2BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x072f	<i>CurrentTier2Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0730	<i>CurrentTier3Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x073N	<i>CurrentTier3BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x073f	<i>CurrentTier3Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0740	<i>CurrentTier4Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x074N	<i>CurrentTier4BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x074f	<i>CurrentTier4Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0750	<i>CurrentTier5Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x075N	<i>CurrentTier5BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x075f	<i>CurrentTier5Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0760	<i>CurrentTier6Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x076N	<i>CurrentTier6BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O



<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Acc</b>	<b>Def</b>	<b>M/O</b>
0x076f	<i>CurrentTier6Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0770	<i>CurrentTier7Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x077N	<i>CurrentTier7BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x077f	<i>CurrentTier7Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0780	<i>CurrentTier8Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x078N	<i>CurrentTier8BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x078f	<i>CurrentTier8Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x0790	<i>CurrentTier9Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x079N	<i>CurrentTier9BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x079f	<i>CurrentTier9Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x007a0	<i>CurrentTier10Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07aN	<i>CurrentTier10BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07af	<i>CurrentTier10Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07b0	<i>CurrentTier11Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07bN	<i>CurrentTier11BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07bf	<i>CurrentTier11Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

Id	Name	Type	Range	Acc	Def	M/O
0x07C0	<i>CurrentTier12Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07cN	<i>CurrentTier12BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07cf	<i>CurrentTier12Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07d0	<i>CurrentTier13Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07dN	<i>CurrentTier13BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07df	<i>CurrentTier13Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07e0	<i>CurrentTier14Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07eN	<i>CurrentTier14BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07ef	<i>CurrentTier14Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07f0	<i>CurrentTier15Block1SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07fN	<i>CurrentTier15BlockN+1SummationDelivered...</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O
0x07ff	<i>CurrentTier15Block16SummationDelivered</i>	uint48	0x000000000000 to 0xFFFFFFFFFFFFFFF	R	-	O

#### 14434 10.4.2.2.8.1 *CurrentTierNBlockNSummationDelivered* Attributes

14435 Attributes *CurrentNoTierBlock1SummationDelivered* through *CurrentTier15Block16SummationDelivered* represent  
 14436 the most recent summed value of Energy, Gas, or Water delivered to the premises (i.e. delivered to the customer  
 14437 from the utility) at a specific price tier as defined by a TOU schedule, Block Threshold or a real time pricing  
 14438 period. If optionally provided, attributes *CurrentNoTierBlock1SummationDelivered* through  
 14439 *CurrentTier15Block16SummationDelivered* are updated continuously as new measurements are made.

14440 **Note:** *SummationFormatting* shall be used against the Block Information attribute set. The expected practical limit  
 14441 for the number of Block attributes supported is 32. The *CurrentTierNBlockNSummationDelivered* attributes are  
 14442 reset at the start of each Block Threshold Period.

14443 **10.4.2.2.9 Alarms Set**

14444 The set of attributes shown in Table 10-48 provides a means to control which alarms may be generated from the  
 14445 meter.

14446 **Note:** Alarms Attribute Set in this revision of this specification is provisional and not certifiable. This feature set  
 14447 may change before reaching certifiable status in a future revision of this specification.

14448 **Table 10-48. Alarm Attribute Set**

Id	Name	Type	Range	Acc	Default	M/O
0x0800	<i>GenericAlarmMask</i>	map16	0x0000 - 0xffff	RW	0xffff	O
0x0801	<i>ElectricityAlarmMask</i>	map32	0x00000000 - 0xffffffff	RW	0xffffffff	O
0x0802	<i>Generic Flow/PressureAlarmMask</i>	map16	0x0000 - 0xffff	RW	0xffff	O
0x0803	<i>WaterSpecificAlarmMask</i>	map16	0x0000 - 0xffff	RW	0xffff	O
0x0804	<i>HeatandCoolingSpecificAlarmMask</i>	map16	0x0000 - 0xffff	RW	0xffff	O
0x0805	<i>GasSpecificAlarmMask</i>	map16	0x0000 - 0xffff	RW	0xffff	O

14449 **10.4.2.2.9.1 AlarmMask Attributes**

14450 The AlarmMask attributes of the Alarm Attribute Set specify whether each of the alarms listed in the corresponding  
 14451 alarm group in Table 10-49 through Table 10-55 is enabled. When the bit number corresponding to the alarm  
 14452 number (minus the group offset) is set to 1, the alarm is enabled, else it is disabled. Bits not corresponding to a  
 14453 code in the respective table are reserved.

14454 **10.4.2.2.9.2 Alarm Codes**

14455 The alarm codes are organized in logical groups corresponding to the meter type as listed in Table 10-49. The three  
 14456 main alarm groups are: Generic, Electricity, and Flow/ Pressure. The Flow/Pressure Alarm Group is further  
 14457 divided into Generic Flow/Pressure, Water Specific, Heat and Cooling Specific, and Gas Specific. It is left for the  
 14458 manufacturer to select which (if any) alarm codes to support.

14459 **Table 10-49. Alarm Code Groups**

Alarm Code	Alarm Condition
00-0F	Generic Alarm Group
10-2F	Electricity Alarm Group
30-7F	Flow/Pressure Alarm Group
30-3F	Generic Flow/Pressure Alarm Group
40-4F	Water Specific Alarm Group
50-5F	Heat and Cooling Specific Alarm Group
60-6F	Gas Specific Alarm Group

14460  
 14461 The generic Alarm Group maps the status from the MeterStatus attribute into a corresponding alarm. Hence,  
 14462 depending on the meter type, an alarm belonging to the Generic Alarm Group may have a different meaning. See  
 14463 sub-clause 10.4.2.2.3. In the case of overlap of alarm codes from the Generic Alarm Group with codes in other

14464 groups, e.g. Burst Detect, it is recommended to only use the code of the Generic Alarm Group, as shown in Table  
14465 10-50.

14466

**Table 10-50. Generic Alarm Group**

Alarm Code	Alarm Condition
00	Check Meter
01	Low Battery
02	Tamper Detect
03	Electricity: Power Failure Gas: Not Defined Water: Pipe Empty Heat/Cooling: Temperature Sensor
04	Electricity: Power Quality Gas: Low Pressure Water: Low Pressure Heat/Cooling: Burst Detect
05	Leak Detect
06	Service Disconnect
07	Electricity: Reserved Gas: Reverse Flow Water: Reverse Flow Heat/Cooling: Flow Sensor
08-0F	Reserved

14467

14468 The Electricity Alarm Group defines alarms specific for electricity meters as defined in Table 10-51.

14469

**Table 10-51. Electricity Alarm Group**

Alarm Code	Alarm Condition
10	Low Voltage L1
11	High Voltage L1
12	Low Voltage L2
13	High Voltage L2
14	Low Voltage L3
15	High Voltage L3
16	Over Current L1
17	Over Current L2
18	Over Current L3
19	Frequency too Low L1

Alarm Code	Alarm Condition
1A	Frequency too High
1B	Frequency too Low L2
1C	Frequency too High
1D	Frequency too Low L3
1E	Frequency too High
1F	Ground Fault
20	Electric Tamper
21-2F	Reserved

14470

14471 The Generic Flow/Pressure Alarm Group defines alarms specific for Flow/Pressure based meters i.e. Water, Heat,  
 14472 Cooling, or Gas meters as defined in Table 10-52.

14473

**Table 10-52. Generic Flow/Pressure Alarm Group**

Alarm Code	Alarm Condition
30	Burst detect
31	Pressure too low
32	Pressure too high
33	Flow sensor communication error
34	Flow sensor measurement fault
35	Flow sensor reverse flow
36	Flow sensor air detect
37	Pipe empty
38-3F	Reserved

14474

14475 The Water Specific Alarm Group defines alarms specific for Water meters as defined in Table 10-53.

14476

**Table 10-53. Water Specific Alarm Group**

Alarm Code	Alarm Condition
40-4F	Reserved

14477

14478 The Heat and Cooling Specific Alarm Group defines alarms specific for Heat or Cooling meters as defined in  
 14479 Table 10-54.

14480

**Table 10-54. Heat and Cooling Specific Alarm Group**

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

14481

14482 The Gas Specific Alarm Group defines alarms specific for Gas meters as defined in Table 10-55.

14483

**Table 10-55. Gas Specific Alarm Group**

Alarm Code	Alarm Condition
60-6F	Reserved

14484 **10.4.2.3 Server Commands**14485 **10.4.2.3.1 Commands Generated**

14486 The command IDs generated by the Metering server cluster are listed in Table 10-56.

14487

**Table 10-56. Generated Command IDs for the Metering Server**

Id	Name	M/O
0x00	<i>Get Profile Response</i>	O
0x01	<i>Request Mirror</i>	O
0x02	<i>Remove Mirror</i>	O
0x03	<i>Request Fast Poll Mode Response</i>	O

14488 **10.4.2.3.1.1 Get Profile Response Command**14489 **10.4.2.3.1.1.1 Payload Format**

14490 The Get Profile Response command payload shall be formatted as illustrated in Figure 10-30.

14491

**Figure 10-30. Format of the Get Profile Response Command Payload**

Octets	4	1	1	1	Variable
Data Type	UTC	enum8	enum8	uint8	Series of uint24s
Field Name	EndTime	Status	ProfileInterval Period	NumberOfPerio dsDelivered	Intervals

14492 **10.4.2.3.1.1.2 Payload Details**14493 **EndTime:** 32-bit value (in UTC) representing the end time of the most chronologically recent interval being  
14494 requested. Example: Data collected from 2:00 PM to 3:00 PM would be specified as a 3:00 PM interval (end

14495 time). It is important to note that the current interval accumulating is not included in most recent block but can  
 14496 be retrieved using the CurrentPartialProfileIntervalValue attribute.

14497 **Status:** Table 10-57 lists the valid values returned in the Status field.

14498 **Table 10-57. Status Field Values**

Value	Description
0x00	Success
0x01	Undefined Interval Channel requested
0x02	Interval Channel not supported
0x03	Invalid End Time
0x04	More periods requested than can be returned
0x05	No intervals available for the requested time

14499  
 14500 **ProfileIntervalPeriod:** Represents the interval or time frame used to capture metered Energy, Gas, and Water  
 14501 consumption for profiling purposes. ProfileIntervalPeriod is an enumerated field representing the timeframes listed  
 14502 in Table 10-58.

14503 **Table 10-58. ProfileIntervalPeriod Timeframes**

Enumerated Value	Timeframe
0	Daily
1	60 minutes
2	30 minutes
3	15 minutes
4	10 minutes
5	7.5 minutes
6	5 minutes
7	2.5 minutes

14504  
 14505 **NumberofPeriodsDelivered:** Represents the number of intervals the device is returning. Please note the number  
 14506 of periods returned in the Get Profile Response command can be calculated when the packets are received and  
 14507 can replace the usage of this field. The intent is to provide this information as a convenience.

14508 **Intervals:** Series of interval data captured using the period specified by the ProfileIntervalPeriod field. The content  
 14509 of the interval data depend of the type of information requested using the Channel field in the Get Profile  
 14510 Command. Data is organized in a reverse chronological order, the most recent interval is transmitted first and the  
 14511 oldest interval is transmitted last. Invalid intervals should be marked as 0xFFFFFFFF.

14512 **10.4.2.3.1.1.3 When Generated**

14513 This command is generated when the Client command GetProfile is received. Please refer to sub-clause  
 14514 10.4.2.4.1.1.

14515 **10.4.2.3.1.2 Request Mirror Command**

14516 This command is used to request the ESI to mirror Metering Device data.

14517 **10.4.2.3.1.2.1 Payload Details**

14518 There are no fields for this command.

14519 **10.4.2.3.1.2.2 Effect on Receipt**

14520 On receipt of this command, the Server shall send a RequestMirrorReponse command (see sub-clause  
14521 10.4.2.4.1.2).

14522 **10.4.2.3.1.3 Remove Mirror Command**

14523 This command is used to request the ESI to remove its mirror of Metering Device data. The device sending the  
14524 Remove Mirror command to the ESI shall send the command to the mirror endpoint to be removed. Only the  
14525 device that created the mirror on the ESI or the ESI itself should be allowed to remove the mirror from the ESI.

14526 **10.4.2.3.1.3.1 Payload Details**

14527 There are no fields for this command.

14528 **10.4.2.3.1.3.2 Effect on Receipt**

14529 On receipt of this command, the Server shall send a MirrorRemoved command (see sub-clause 10.4.2.4.1.3).

14530 **10.4.2.3.1.4 Request Fast Poll Mode Response Command**

14531 **10.4.2.3.1.4.1 Payload Format**

14532 The Request Fast Poll Mode Response command payload shall be formatted as illustrated in Figure 10-31.

14533 **Figure 10-31. Format of the Request Fast Poll Mode Response Command Payload**

<b>Octets</b>	1	4
<b>Data Type</b>	uint8	UTC
<b>Field Name</b>	Applied Update Period (seconds) (M)	Fast Poll Mode End Time (M)

14534 **10.4.2.3.1.4.2 Payload Details**

14535 **Applied Update Period:** The period at which metering data shall be updated. This may be different than the  
14536 requested fast poll. If the Request Fast Poll Rate is less than Fast Poll Update Period Attribute, it shall use the  
14537 Fast Poll Update Period Attribute. Otherwise, the Applied Update Period shall be greater than or equal to the  
14538 minimum Fast Poll Update Period Attribute and less than or equal to the Requested Fast Poll Rate.

14539 **Fast Poll Mode End Time:** UTC time that indicates when the metering server will terminate fast poll mode and  
14540 resume updating at the rate specified by DefaultUpdatePeriod. For example, one or more metering clients may  
14541 request fast poll mode while the metering server is already in fast poll mode. The intent is that the fast poll mode will  
14542 not be extended since this scenario would make it possible to be in fast poll mode longer than 15 minutes.

14543 **10.4.2.3.1.4.3 When Generated**

14544 This command is generated when the client command Request Fast Poll Mode is received.

14545 **10.4.2.3.1.4.4 Effect on Receipt**

14546 On receipt of this command, the device may request or receive updates not to exceed the Applied Update Period  
14547 until Fast Poll Mode End Time.



14548 **10.4.2.4 Client Commands**

14549 **10.4.2.4.1 Commands Generated**

14550 The command IDs generated by the Metering client cluster are listed in Table 10-59.

14551 **Table 10-59. Generated Command IDs for the Metering Client**

Command Identifier Field Value	Description	M/O
0x00	<i>Get Profile</i>	O
0x01	<i>Request Mirror Response</i>	O
0x02	<i>Mirror Removed</i>	O
0x03	<i>Request Fast Poll Mode</i>	O

14552 **10.4.2.4.1.1 Get Profile Command**

14553 The Get Profile command payload shall be formatted as illustrated in Figure 10-32.

14554 **Figure 10-32. Format of the Get Profile Command Payload**

<b>Octets</b>	1	4	1
<b>Data Type</b>	enum8	UTC	uint8
<b>Field Name</b>	Interval Channel	End Time	NumberOfPeriods

14555 **10.4.2.4.1.1.1 Payload Details**

14556 **Interval Channel:** Enumerated value used to select the quantity of interest returned by the GetProfileResponse command. The Interval Channel values are listed in Table 10-60.

14558 **Table 10-60. Interval Channel Values**

Enumerated Value	Description
0	Consumption
1	Consumption Received

14559  
 14560 **EndTime:** 32-bit value (in UTC) used to select an Intervals block from all the Intervals blocks available. The  
 14561 Intervals block returned is the most recent block with its EndTime equal or older to the one provided. The most  
 14562 recent Intervals block is requested using an End Time set to 0x00000000, subsequent Intervals block are requested  
 14563 using an End time set to the EndTime of the previous block - (number of intervals of the previous block \*  
 14564 ProfileIntervalPeriod).

14565 **NumberOfPeriods:** Represents the number of intervals being requested. This value can't exceed the size  
 14566 stipulated in the MaxNumberOfPeriodsDelivered attribute. If more intervals are requested than can be delivered,  
 14567 the GetProfileResponse will return the number of intervals equal to MaxNumberOfPeriodsDelivered. If fewer  
 14568 intervals are available for the time period, only those available are returned.

14569 **10.4.2.4.1.1.2 When Generated**

14570 The GetProfile command is generated when a client device wishes to retrieve a list of captured Energy, Gas or water  
14571 consumption for profiling purposes. Due to the potentially large amount of profile data available, the client device  
14572 should store previously gathered data and only request the most current data. When initially gathering significant  
14573 amounts of historical interval data, the GetProfile command should not be issued any more frequently than 7.5  
14574 seconds to prevent overwhelming the ZigBee network.

#### 14575 **10.4.2.4.1.1.3 Command Processing Response**

14576 If failure occurs in recognizing or processing the payload of the GetProfile command, the appropriate  
14577 enumerated ZCL status (as defined in Chapter 2) will be returned. On success, a non-Default Response is returned  
14578 without a ZCL status code.

#### 14579 **10.4.2.4.1.1.4 Effect on Receipt**

14580 On receipt of this command, the device shall send a GetProfileReponse command (see sub-clause 10.4.2.3.1.1).

#### 14581 **10.4.2.4.1.2 Request Mirror Response Command**

14582 The Request Mirror Response Command allows the ESI to inform a sleepy Metering Device it has the ability to  
14583 store and mirror its data.

#### 14584 **10.4.2.4.1.2.1 Payload Format**

14585 The Request Mirror Response command payload shall be formatted as illustrated in Figure 10-33.

14586 **Figure 10-33. Format of the Request Mirror Response Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	EndPoint ID

#### 14587 **10.4.2.4.1.2.2 Payload Details**

14588 **EndPoint ID:** 16 Bit Unsigned Integer indicating the End Point ID to contain the Metering Devices meter data.  
14589 Valid End Point ID values are 0x0001 to 0x00F0. If the ESI is able to mirror the Metering Device data, the low byte  
14590 of the unsigned 16 bit integer shall be used to contain the eight bit EndPoint ID. If the ESI is unable to mirror the  
14591 Metering Device data, EndPoint ID shall be returned as 0xFFFF. All other EndPoint ID values are reserved. If  
14592 valid, the Metering device shall use the EndPoint ID to forward its metered data.

#### 14593 **10.4.2.4.1.3 Mirror Removed Command**

14594 The Mirror Removed Command allows the ESI to inform a sleepy Metering Device mirroring support has been  
14595 removed or halted.

#### 14596 **10.4.2.4.1.3.1 Payload Format**

14597 The Mirror Removed command payload shall be formatted as illustrated in Figure 10-34.

14598 **Figure 10-34. Format of the Mirror Removed Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Removed EndPoint ID

#### 14599 **10.4.2.4.1.3.2 Payload Details**

14600 **Removed EndPoint ID:** 16 Bit Unsigned Integer indicating the End Point ID previously containing the Metering  
14601 Device's meter data.

14602 **10.4.2.4.1.4 Request Fast Poll Mode Command**

14603 **10.4.2.4.1.4.1 Payload Format**

14604 The Request Fast Poll Mode shall be formatted as illustrated in Figure 10-35.

14605 **Figure 10-35. Format of the Request Fast Poll Mode Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	Fast Poll Update Period (seconds)	Duration (minutes)

14606 **10.4.2.4.1.4.2 Payload Details**

14607 **Fast Poll Update Period:** Desired fast poll period not to be less than the FastPollUpdatePeriod attribute.

14608 **Duration:** Desired duration for the server to remain in fast poll mode not to exceed 15 minutes as specified in  
 14609 sub-clause 10.4.3.2.

14610 **10.4.2.4.1.4.3 When Generated**

14611 The Request Fast Poll Mode command is generated when the metering client wishes to receive near real-time  
 14612 updates of InstantaneousDemand. Fast poll mode shall only be requested as a result of user interaction (for example,  
 14613 the pushing of a button or activation of fast poll mode by a menu choice).

14614 **10.4.2.4.1.4.4 Effect on Receipt**

14615 The metering device may continuously update InstantaneousDemand as measurements are acquired, but at a  
 14616 minimum InstantaneousDemand must be updated at the end of each FastPollUpdatePeriod.

14617 **10.4.3 Metering Application Guidelines**

14618 **10.4.3.1 Attribute Reporting**

14619 Attribute reporting may be used for sending information in the Reading Information, TOU Information, Meter  
 14620 Status, and Historical Consumption attribute sets. Use of the Report Attribute command without report configuration  
 14621 may be used for unsolicited notification of an attribute value change. Sleepy devices may have to poll.

14622 **10.4.3.2 Fast Polling or Reporting for Monitoring Energy Savings**

14624 Client devices, such as an energy gateway, smart thermostat, or in-home displays can monitor changes to energy  
 14625 saving settings within the premises and give users near real time feedback and results. The Metering cluster can  
 14626 support this by using Attribute Reporting and sending updates at a much faster rate for a short period of time.  
 14627 Client devices can also perform a series of Attribute reads to accomplish the same task. In either case, requests  
 14628 or updates shall be limited to a maximum rate of once every two seconds for a maximum period of 15 minutes.  
 14629 These limitations are required to ensure Smart Energy profile based devices do not waste available bandwidth or  
 14630 prevent other operations within the premises.

14631 **10.4.3.3 Metering Data Updates**

14632 The frequency and timeliness of updating metering data contained in the Metering Cluster attributes and Profile  
 14633 Intervals is up to the individual Metering device manufacturer’s capabilities. As a best practice recommendation,  
 14634 updates of the metering data should not cause delivery of the information to end devices more often than once  
 14635 every 30 seconds. End devices should also not request information more often than once every 30 seconds.

14636 The Fast Polling attributes and commands shall be used by client devices requesting information more often than  
14637 once every 30 seconds.

#### 14638 **10.4.3.3.1 Fast Polling Periods**

14639 Since the DefaultUpdatePeriod specifies the normal update interval and FastPollUpdatePeriod specifies the fastest  
14640 possible update interval, it is recommended that metering clients read these attributes to determine the optimal  
14641 normal/fast polling interval and the optimal fast poll period to request. Client devices shall not request data more  
14642 frequent than FastPollUpdatePeriod or the AppliedUpdatePeriod.

#### 14643 **10.4.3.4 Mirroring**

14644 SE Profile specifies Mirror support in the Metering cluster to store and provide access to data from metering  
14645 devices on battery power. Devices with resources to support mirroring advertize the capability using the Basic  
14646 Attribute Physical Environment.

#### 14647 **10.4.3.4.1 Discovery**

14648 The SE standard does not prescribe how Mirroring is implemented. Devices may query the Basic Cluster attribute  
14649 PhysicalEnvironment to determine Mirrored device capacity prior to CBKE (see sub-clause 10.4.3.4.2). This  
14650 would allow a battery based end device to discover if an ESI has capacity to mirror data prior to the process of  
14651 joining the network in a secure manner, thereby reducing retry attempts. This would also enhance the service  
14652 discovery of the ZDO Match Descriptor that would be used to determine if an endpoint can request the setup and  
14653 removal of a mirrored Metering cluster. Once a device has joined the network and performed CBKE, it can  
14654 then request setup of a mirrored metering cluster. ZDO Discovery should be supported to allow HAN devices to  
14655 discover the mirror endpoints; only active mirror endpoints shall be discoverable.

#### 14656 **10.4.3.4.2 Mirror Attributes**

14657 The mandatory Basic, Metering, and (where applicable) Prepayment attributes shall be supported. The Basic  
14658 Cluster PhysicalEnvironment attribute shall be supported on ESIs supporting mirroring functionality; an  
14659 enumerated value of 0x01 would indicate that the device has the capacity to mirror an end device; a value of  
14660 0x00 would specify an “Unspecified environment” per the ZCL specification. Only the Basic cluster for devices  
14661 capable of providing a mirror shall have the PhysicalEnvironment attribute set to 0x01. The ZCL Report Attribute  
14662 command shall be used to push data to the mirror. Only the metering device that has been granted a mirror on a  
14663 certain endpoint is allowed to push data to that endpoint. The ZCL Not Authorized return status shall be used to  
14664 provide access control. The use of ZCL Report Configuration shall not be required to generate Report Attribute  
14665 Command.

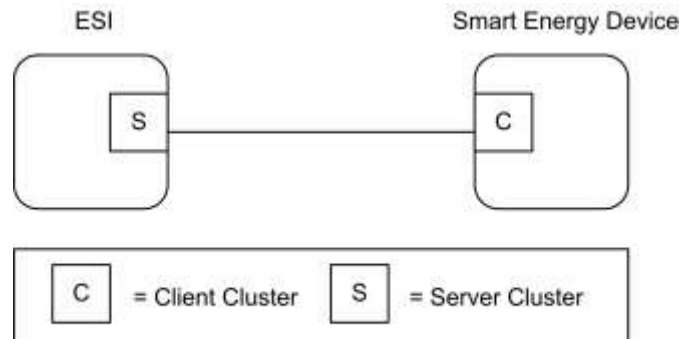
14666 Manufacturers will design and manufacture devices to meet customer requirement specifications that will state the  
14667 functionality of the battery powered meter and therefore devices supporting mirroring in the field will also have to  
14668 support those requirements through an appropriate choice of optional attributes. Battery powered devices will report  
14669 attributes to the mirror as required by the customer specification. In the event that the mirror is out of memory  
14670 space or cannot support the attribute it shall respond ATTRIBUTE\_UNSUPPORTED back to the battery-powered  
14671 meter. The same response (ATTRIBUTE\_UNSUPPORTED) will be sent to a device querying the mirror for an  
14672 attribute it doesn't support. A device querying the mirror for an attribute that is supported but not yet available  
14673 (the battery powered meter hasn't yet sent the attribute) shall receive a response  
14674 ATTRIBUTE\_UNAVAILABLE from the mirror.

14675 **10.5 Messaging**

14676 **10.5.1 Overview**

14677 This cluster provides an interface for passing text messages between devices. Messages are expected to be  
 14678 delivered via the ESI and then unicast to all individually registered devices implementing the Messaging Cluster  
 14679 on the network, or just made available to all devices for later pickup. Nested and overlapping messages are not  
 14680 allowed. The current active message will be replaced if a new message is received by the ESI.

14681 **Figure 10-36. Messaging Cluster Client/Server Example**



*Note: Device names are examples for illustration purposes only*

14682 Please note the ESI is defined as the Server due to its role in acting as the proxy for upstream message management  
 14683 systems and subsequent data stores.  
 14684

14685 **10.5.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

14686 **10.5.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	SEMS	Type 1 (client to server)

14687 **10.5.1.3 Cluster Identifiers**

Identifier	Name
0x0703	Messaging (Smart Energy)

14688 **10.5.2 Server**

14689 **10.5.2.1 Dependencies**

14690 Support for ZCL Data Types.

- 14691 Anonymous Inter-PAN transmission mechanism.  
14692 No dependencies exist for other Smart Energy Clusters.

### 14693 10.5.2.2 Attributes

14694 None

### 14695 10.5.2.3 Commands Generated

14696 The command IDs generated by the Messaging server cluster are listed in Table 10-61.

14697 **Table 10-61. Generated Command IDs for the Messaging Server**

Command Identifier Field Value	Description	M/O
0x00	<i>Display Message</i>	M
0x01	<i>Cancel Message</i>	M

#### 14698 10.5.2.3.1 Display Message Command

##### 14699 10.5.2.3.1.1 Payload Format

14700 The Display Message command payload shall be formatted as illustrated in Figure 10-37.

14701 **Figure 10-37. Format of the Display Message Command Payload**

<b>Octets</b>	4	1	4	2	Variable
<b>Data Type</b>	uint32	map8	UTC	uint16	string
<b>Field Name</b>	Message ID	Message Control	Start Time	Duration In Minutes	Message

##### 14702 10.5.2.3.1.1.1 Payload Details

14703 **Message ID:** A unique unsigned 32-bit number identifier for this message. It's expected the value contained in  
14704 this field is a unique number managed by upstream systems or a UTC based time stamp (UTC data type)  
14705 identifying when the message was issued.

14706 **MessageControl:** An 8-bit bitmap field indicating the need to optionally pass the message onto the Anonymous  
14707 Inter-PAN transmission mechanism or that a user confirmation is required for a message. Bit encoding of this field is  
14708 outlined in Table 10-62.

14709 **Table 10-62. Message Control Field Bit Map**

Bits	Enumeration	Value	Description
0 to 1	Normal transmission only	0	Send message through normal command function to client.
	Normal and Anonymous Inter- PAN transmission	1	Send message through normal command function to client and pass message onto the Anonymous Inter-PAN transmission mechanism.

Bits	Enumeration	Value	Description
	Anonymous Inter- PAN transmission only	2	Send message through the Anonymous Inter- PAN transmission mechanism.
	<i>Reserved</i>	3	Reserved value for future use.
2 to 3	Low	0	Message to be transferred with a low level of importance.
	Medium	1	Message to be transferred with a medium level of importance.
	High	2	Message to be transferred with a high level of importance.
	Critical	3	Message to be transferred with a critical level of importance.
4 to 6	<i>Reserved</i>	<i>N/A</i>	These bits are reserved for future use.
7	Message Confirmation	0	Message Confirmation not required.
		1	Message Confirmation required.

14710

14711 If the Anonymous Inter-PAN transmission mechanism is not supported on a particular device, Bits 0 to 6 can be  
 14712 ignored.

14713 The Message Confirmation bit indicates the message originator requests a confirmation of receipt from a Utility  
 14714 Customer. If confirmation is required, the device should display the message or alert the user until it is either  
 14715 confirmed via a button, by selecting a confirmation option on the device, or the message expires. Confirmation is  
 14716 typically used when the Utility is sending down information such as a disconnection notice, or prepaid billing  
 14717 information.

14718 **Note:** It is desired that the device provide a visual indicator (flashing display or indicate with its LEDs as  
 14719 examples) that a message requiring confirmation is being displayed, and requires confirmation.

14720 **Start Time:** A UTC field to denote the time at which the message becomes valid. A Start Time of 0x00000000 is a  
 14721 special time denoting “now.” If the device would send an event with a Start Time of now, adjust the Duration In  
 14722 Minutes field to correspond to the remainder of the event.

14723 **Duration In Minutes:** An unsigned 16-bit field is used to denote the amount of time in minutes after the Start  
 14724 Time during which the message is displayed. A Maximum value of 0xFFFF means “until changed”.

14725 **Message:** A ZCL String containing the message to be delivered. The String shall be encoded in the UTF-8 format.  
 14726 Please note: Since the Anonymous Inter-PAN transmission mechanism does not support fragmentation and is limited  
 14727 in its message size, any message forwarded will be truncated to match the maximum message length supported.  
 14728 For messages sent through the Anonymous Inter-PAN transmission mechanism and received by devices that  
 14729 display messages smaller than 80 bytes, they shall have the ability to receive up to an 80 byte message.  
 14730 Devices will have the ability to choose the methods for managing messages that are larger than can be displayed  
 14731 (truncation, scrolling, etc.).

14732 For supporting larger messages sent over the SE Profile network, both devices must agree upon a common  
 14733 Fragmentation ASDU Maximum Incoming Transfer Size. Please refer to [Z1] for further details on Fragmentation  
 14734 settings.

14735 Any message that needs truncation shall truncate on a UTF-8 character boundary. The SE secure payload is 59 bytes  
 14736 for the Message field in a non- fragmented, non-source routed Display Message packet (11 bytes for other Display  
 14737 Message fields). Devices using fragmentation can send a message larger than this. Reserving bytes for source route  
 14738 will reduce this. InterPAN message payload for the “message” is 98 bytes.

14739 **10.5.2.3.2 Cancel Message Command**

14740 The Cancel Message command described in Figure 10-38 provides the ability to cancel the sending or acceptance  
14741 of previously sent messages. When this message is received the recipient device has the option of clearing any display  
14742 or user interfaces it supports, or has the option of logging the message for future reference.

14743 **Figure 10-38. Format of the Cancel Message Command Payload**

<b>Octets</b>	4	1
<b>Data Type</b>	uint32	map8
<b>Field Name</b>	Message ID	Message Control

14744 **10.5.2.3.2.1 Payload Details**

14745 **Message ID:** A unique unsigned 32-bit number identifier for the message being cancelled. It's expected the  
14746 value contained in this field is a unique number managed by upstream systems or a UTC based time stamp  
14747 (UTC data type) identifying when the message was originally issued.

14748 **MessageControl:** An enumerated field indicating the optional ability to pass the cancel message request onto the  
14749 Anonymous Inter-PAN transmission mechanism. If the Anonymous Inter-PAN transmission mechanism is not  
14750 supported on a particular device, this parameter is ignored. Bitmap values for this field are listed in Table 10-62.

14751 **10.5.3 Client**14752 **10.5.3.1 Dependencies**

14753 Support for ZCL Data Types.

14754 No dependencies exist for other Smart Energy Clusters.

14755 **10.5.3.2 Attributes**

14756 None

14757 **10.5.3.3 Commands Generated**

14758 The command IDs generated by the Messaging cluster are listed in Table 10-63.

14759 **Table 10-63. Messaging Client Commands**

<b>Id</b>	<b>Description</b>	<b>M/O</b>
0x00	<i>Get Last Message</i>	M
0x01	<i>Message Confirmation</i>	M

14760 **10.5.3.3.1 GetLastMessage Command**

14761 This command has no payload.

14762 **10.5.3.3.1.1 Effect on Receipt**

14763 On receipt of this command, the device shall send a Display Message command (refer to sub-clause 10.5.2.3.1). A  
14764 ZCL Default Response with status NOT\_FOUND shall be returned if no message is available.



14765 **10.5.3.3.2 MessageConfirmation Command**

14766 The Message Confirmation command described in Figure 10-39 provides the ability to acknowledge a previously  
14767 sent message.

14768 **Figure 10-39. Format of the *Message Confirmation* Command Payload**

<b>Octets</b>	4	4
<b>Data Type</b>	uint32	UTC
<b>Field Name</b>	Message ID	Confirmation Time

14769 **10.5.3.3.2.1 Payload Details**

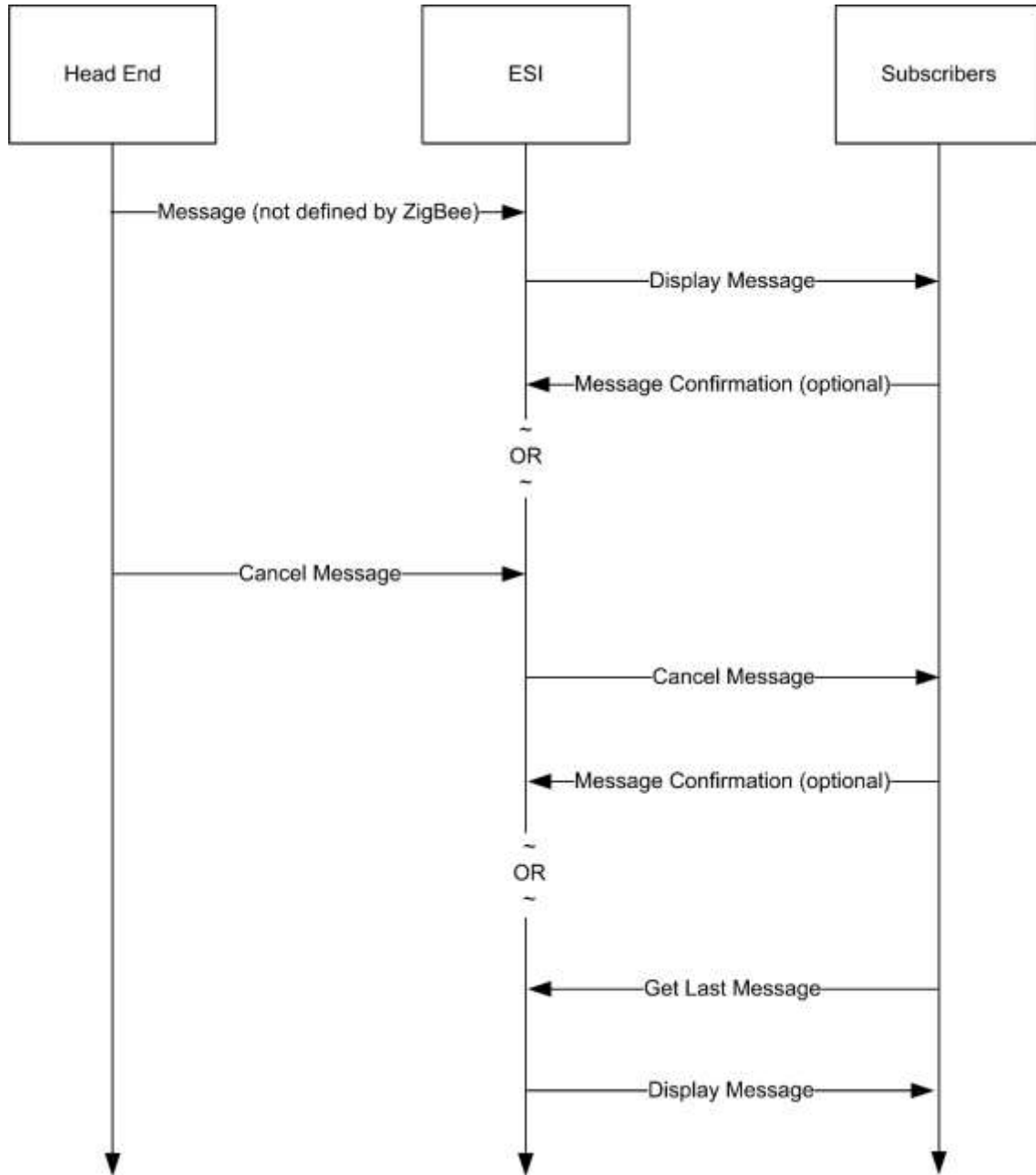
14770 **Message ID:** A unique unsigned 32-bit number identifier for the message being confirmed.

14771 **Confirmation Time:** UTC of user confirmation of message.

14772 **10.5.4 Application Guidelines**

14773 For Server and Client transactions, refer to Figure 10-40.

14774 **Figure 10-40. Client/Server Message Command Exchanges**



14775

14776 **10.6 Tunneling**

14777 **Note:** The optional support for flow control within the cluster in this revision of this specification is provisional  
 14778 and not certifiable. This feature set may change before reaching certifiable status in a future revision of this  
 14779 specification.

## 10.6.1 Overview

14780

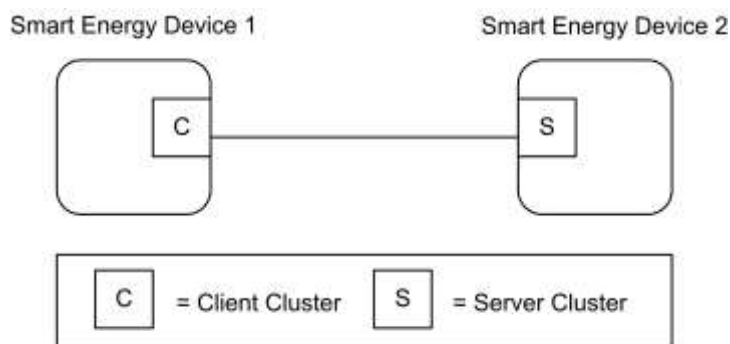
14781 The tunneling cluster provides an interface for tunneling protocols. It is comprised of commands and attributes  
 14782 required to transport any existing metering communication protocol within the payload of standard ZigBee frames  
 14783 (including the handling of issues such as addressing, fragmentation and flow control). Examples for such protocols  
 14784 are DLMS/COSEM, IEC61107, ANSI C12, M-Bus, ClimateTalk, etc.

14785 The tunneling foresees the roles of a server and a client taking part in the data exchange. Their roles are defined  
 14786 as follows:

14787 **Client:** Requests a tunnel from the server and closes the tunnel if it is no longer needed.

14788 **Server:** Provides and manages tunnels to the clients.

14789 **Figure 10-41. A Client Requests a Tunnel from a Server to Exchange Complex Data in Both Directions**



*Note: Device names are examples for illustration purposes only*

14790

14791

14792 The data exchange through the tunnel is symmetric. This means both client and server provide the commands to  
 14793 transfer data (TransferData). And both must make sure that only the partner to which the tunnel has been built up is  
 14794 granted RW access to it (e.g. tunnel identifier protection through checking the MAC address).

14795 Sleepy devices either close the tunnel immediately after they have pushed their data through it, or leave it open in  
 14796 which case an attribute in the server (CloseTunnelTimeout) decides whether the tunnel is closed from the server  
 14797 side during the sleeping phase or not. If data is transferred to a non-existent or wrong tunnel identifier, the receiver  
 14798 generates an error message (TransferDataError).

14799 The server may support more than one tunneling protocol. The type of tunnel to be opened is a mandatory parameter  
 14800 (ProtocolID) of the tunnel request (RequestTunnel) that the client needs to send to the server in order to set up a  
 14801 new tunnel. The response from the server (RequestTunnelResponse) will contain a parameter with the status of  
 14802 the tunnel (TunnelStatus). If the tunnel request was successful, a unique identifier (TunnelID) is returned within the  
 14803 response. In an error case (e.g. the requested protocol is not supported) the status contains the type of error. There  
 14804 is no special attribute in order to read out the supported protocols from the server. Either the client knows them a  
 14805 priori or it has to try several times using different ProtocolIDs until the server responds with the tunnel status  
 14806 Success.

14807 The tunneling cluster adds optional support for flow control to handle streaming protocols such as IEC61107. If  
 14808 implemented, flow control messages are provided to control the data flow and send acknowledges to data messages  
 14809 on application level. However, flow control is an optional feature and disabled per default. In the default case, the  
 14810 acknowledge messages (AckTransferData) must not be sent in order to reduce complexity and prevent from  
 14811 unneeded overhead.

14812 The following sequence describes a typical usage:

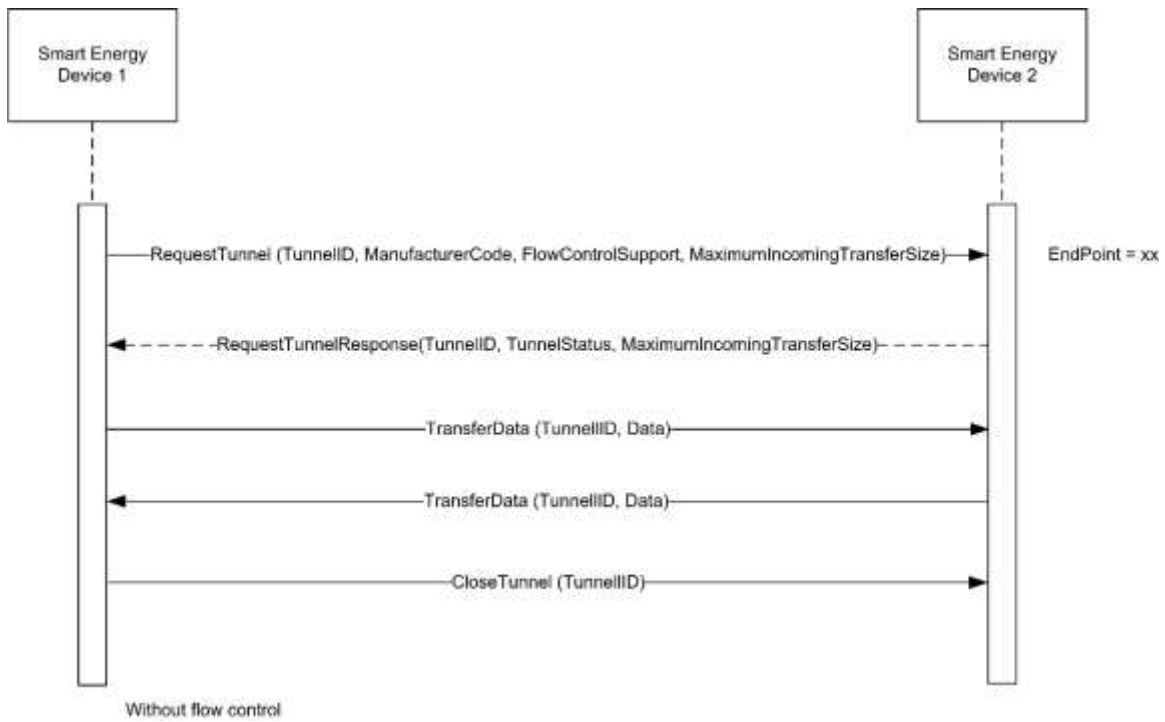
1. 14813 The client issues a service discovery to find devices which support the tunneling server cluster. The  
 14814 discovery may either be directed to one device, if its address is known, or be a broadcast  
 14815 (*MatchSimpleDescriptor*).

- 14816 2. The response to the discovery from the server contains an endpoint number (*SimpleDescriptor*). Using this  
14817 endpoint, the client directs a tunnel request to a given server. Together with the request, the client is required  
14818 to provide an enumeration with the ID of the protocol that shall be tunneled. There is the possibility to  
14819 request tunnels for manufacturer specific protocols. In this case, the *ProtocolID* has to be followed by a  
14820 *ZigBee ManufacturerCode* to open the tunnel. An additional parameter for *FlowControlSupport*  
14821 accompanies the request, together with an indication of the client's incoming buffer size (*RequestTunnel*  
14822 (*ProtocolID, ManufacturerCode, FlowControlSupport, MaximumIncomingTransferSize*)).
- 14823 3. If the server supports the protocol, it allocates the required resources, assigns a tunnel identifier and returns  
14824 the ID number within the response including an additional tunnel status that the command was successful  
14825 and the server's incoming buffer size. If the command failed, the status contains the reason in form of an  
14826 error code (*RequestTunnelResponse (TunnelID, TunnelStatus, MaximumIncomingTransferSize)*). The  
14827 tunnel identifier number would then be invalid in this case.
- 14828 4. Both server and client may exchange data (*TransferData(Data)*). In case the optional flow control is  
14829 utilized, each data transfer is acknowledged (*AckTransferData(NumberOfOctetsLeft)*). Additionally, there is  
14830 the possibility to stop (*AckTransferData(0)*) and resume (*ReadyData(NumberOfOctetsLeft)*) the data  
14831 transfer.
- 14832 5. After the transfer has been successfully completed, the client closes the tunnel again freeing the tunnel  
14833 identifier in the server (*CloseTunnel(TunnelID)*). If not, the server closes the tunnel by itself after  
14834 *CloseTunnelTimeout* seconds.

14835 The following sequence diagrams show the client/server model and the typical usage of the cluster without flow  
14836 control (Figure 10-42) and with flow control (Figure 10-43).

14837  
14838

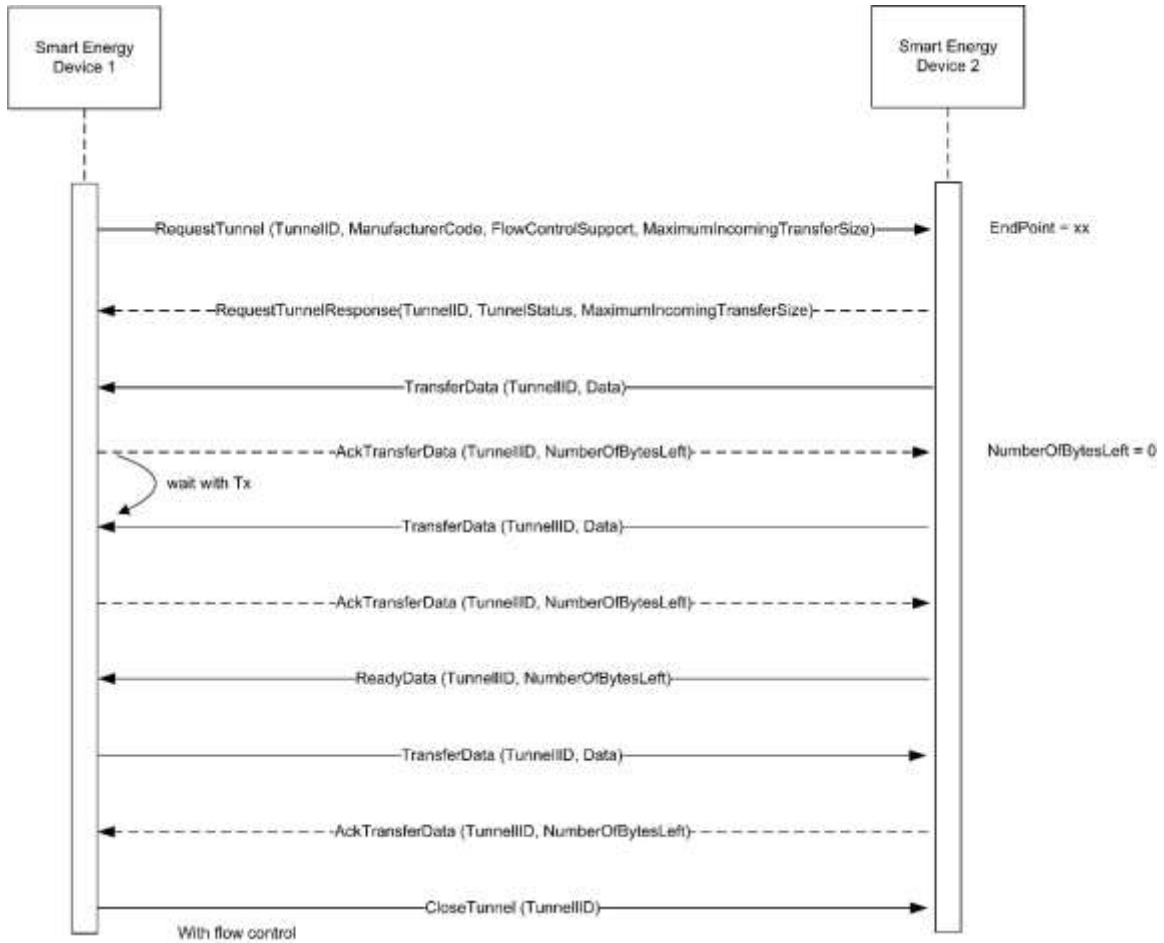
**Figure 10-42. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data Without Flow Control (Default)**



14839

14840

**Figure 10-43. SE Device 1 (Client) Requests a Tunnel from SE Device 2 (Server) to Transfer Data with Flow Control**



### 10.6.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 10.6.1.2 Classification

Hierarchy	Role	PICS Code
Base	Application	SETUN

### 10.6.1.3 Cluster Identifiers

Identifier	Name
0x0704	Tunneling (Smart Energy)

14845 **10.6.2 Server**

14846 **10.6.2.1 Dependencies**

14847 This cluster requires APS fragmentation [Z1] to be implemented, with maximum transfer sizes defined by the  
 14848 device's negotiated input buffer sizes.

14849 **10.6.2.2 Attributes**

14850 **Table 10-64. Tunneling Cluster Attributes**

Identifier	Name	Type	Range	Access	Default	M/O
0x0000	<i>CloseTunnelTimeout</i>	uint16	0x0001-0xFFFF	R	0xFFFF	M

14851 **10.6.2.2.1 CloseTunnelTimeout Attribute**

14852 CloseTunnelTimeout defines the minimum number of seconds that the server waits on an inactive tunnel before  
 14853 closing it on its own and freeing its resources (without waiting for the CloseTunnel command from the client).  
 14854 Inactive means here that the timer is re-started with each new reception of a command. 0x0000 is an invalid value.

14855 **10.6.2.3 Parameters**

14856 Table 10-65 contains a summary of all parameters passed to or returned by the server commands. These values are  
 14857 considered as parameters (and not attributes) in order to facilitate the handling of the tunneling cluster for both the  
 14858 client and the server side. The parameters cannot be read or written via ZCL global commands. The detailed  
 14859 description of these parameters can be found in the according command sections of the document.

14860 **Table 10-65. Cluster Parameters Passed Through Commands**

Name	Type	Range	Default	M/O
ProtocolID	enum8	0x01 – 0xFF	0x00	M
ManufacturerCode	uint16	0x0000 – 0xFFFF	0x00	M
FlowControlSupport	Boolean	TRUE or FALSE	FALSE	M
MaximumIncomingTransferSize	uint16	0x0000 – 0xFFFF	1500	M
TunnelID	uint16	0x0000 – 0xFFFF	(Return value)	M
Data	octstr	-	-	M
NumberOfOctetsLeft	uint16	0x0000 – 0xFFFF	-	M
TunnelStatus	uint8	0x00 – 0x04	-	M
TransferDataStatus	uint8	0x00 – 0x01	-	M

14861 **10.6.2.4 Commands Received**

14862 Table 10-66 lists cluster-specific commands received by the server.

14863

**Table 10-66. Cluster-specific Commands Received by the Server**

Command Identifier Field Value	Description	M/O
0x00	<i>RequestTunnel</i>	M
0x01	<i>CloseTunnel</i>	M
0x02	<i>TransferData</i>	M
0x03	<i>TransferDataError</i>	M
0x04	<i>AckTransferData</i>	O
0x05	<i>ReadyData</i>	O
0x06	<i>GetSupportedTunnelProtocols</i>	O

14864 **10.6.2.4.1 RequestTunnel Command**

14865 RequestTunnel is the client command used to setup a tunnel association with the server. The request payload  
 14866 specifies the protocol identifier for the requested tunnel, a manufacturer code in case of proprietary protocols and the  
 14867 use of flow control for streaming protocols.

14868 **10.6.2.4.1.1 Payload Format**14869 **Figure 10-44. Format of the RequestTunnel Command Payload**

Octets	1	2	1	2
<b>Data Type</b>	enum8	uint16	bool	uint16
<b>Field Name</b>	ProtocolID (M)	Manufacturer Code (M)	FlowControl Support (M)	Maximum Incoming TransferSize

14870 **10.6.2.4.1.2 Payload Details**

14871 **ProtocolID:** An enumeration representing the identifier of the metering communication protocol for which the  
 14872 tunnel is requested. Table 10-67 lists the possible values for the ProtocolID. The values above 199 may be used for  
 14873 manufacturer-specific protocols.

14874 **Table 10-67. ProtocolID Enumerations**

Values	Description
0	DLMS/COSEM (IEC 62056)
1	IEC 61107
2	ANSI C12
3	M-BUS
4	SML
5	ClimateTalk
200 to 254	Manufacturer-defined protocols

14875



14876 **Manufacturer Code:** A code that is allocated by the ZigBee Alliance, relating the manufacturer to a device and – for  
 14877 the tunneling - a manufacturer specific protocol. The parameter is ignored when the ProtocolID value is less  
 14878 than 200. This allows for 55 manufacturer-defined protocols for each manufacturer to be defined. A value of  
 14879 0xFFFF indicates that the Manufacturer Code is not used.

14880 **FlowControlSupport:** A Boolean type parameter that indicates whether flow control support is requested from  
 14881 the tunnel (TRUE) or not (FALSE). The default value is FALSE (no flow control).

14882 **MaximumIncomingTransferSize:** A value that defines the size, in octets, of the maximum data packet that can  
 14883 be transferred to the client in the payload of a single TransferData command.

14884 **10.6.2.4.1.3 When Generated**

14885 Is never generated by the server.

14886 **10.6.2.4.1.4 Effect on Receipt**

14887 Triggers a process within the server to allocate resources and build up a new tunnel. A RequestTunnelResponse  
 14888 is generated and sent back to the client containing the result of the RequestTunnel command.

14889 **10.6.2.4.2 CloseTunnel Command**

14890 Client command used to close the tunnel with the server. The parameter in the payload specifies the tunnel identifier  
 14891 of the tunnel that has to be closed. The server leaves the tunnel open and the assigned resources allocated until the  
 14892 client sends the CloseTunnel command or the CloseTunnelTimeout fires.

14893 **10.6.2.4.2.1 Payload Format**

14894 **Figure 10-45. Format of the *CloseTunnel* Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	TunnelID (M)

14895 **10.6.2.4.2.2 Payload Details**

14896 **TunnelID:** The identifier of the tunnel that shall be closed. It is the same number that has been previously returned  
 14897 in the response to a RequestTunnel command. Valid numbers range between 0..65535 and must correspond to a  
 14898 tunnel that is still active and maintained by the server.

14899 **10.6.2.4.2.3 When Generated**

14900 This command is never generated by the server.

14901 **10.6.2.4.2.4 Effect on Receipt**

14902 In case the given TunnelID is correct, the server closes the tunnel and frees the resources. The associated tunnel  
 14903 is no longer maintained.

14904 **10.6.2.4.3 TransferData Command**

14905 Command that indicates (if received) that the client has sent data to the server. The data itself is contained within  
 14906 the payload.

14907 **10.6.2.4.3.1 Payload Format**

14908

**Figure 10-46. Format of the *TransferData* Command Payload**

<b>Octets</b>	2	Variable
<b>Data Type</b>	uint16	opaque
<b>Field Name</b>	TunnelID (M)	Data (M)

14909 **10.6.2.4.3.2 Payload Details**

14910 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
 14911 triggered through the RequestTunnel command. This ID must be used to send data through the tunnel or passed with  
 14912 any commands concerning that specific tunnel.

14913 **Data:** Series of octets containing the data to be transferred through the tunnel in the format of the communication  
 14914 protocol for which the tunnel has been requested and opened. The payload contains the assembled data exactly as it  
 14915 was sent by the client. Theoretically, its length is solely limited through the fragmentation algorithm and the RX/TX  
 14916 transfer buffer sizes within the communication partners. The content of the payload is up to the application sending  
 14917 the data. It is neither guaranteed, that it contains a complete PDU nor is any other assumption on its internal format  
 14918 made. This is left up to the implementer of the specific protocol tunnel behavior.

14919 **10.6.2.4.3.3 When Generated**

14920 Is generated whenever the server wants to tunnel protocol data to the client.

14921 **10.6.2.4.3.4 Effect on Receipt**

14922 Indicates that the server has received tunneled protocol data from the client.

14923 **10.6.2.4.4 TransferDataError Command**

14924 This command is generated by the receiver of a TransferData command if the tunnel status indicates that something  
 14925 is wrong. There are three cases in which TransferDataError is sent:

- 14926 • The *TransferData* received contains a *TunnelID* that does not match to any of the active tunnels of the  
 14927 receiving device. This could happen if a (sleeping) device sends a *TransferData* command to a tunnel that  
 14928 has been closed by the server after the *CloseTunnelTimeout*.
- 14929 • The *TransferData* received contains a proper *TunnelID* of an active tunnel, but the device sending the data  
 14930 does not match to it.
- 14931 • The *TransferData* received contains more data than indicated by the *MaximumIncomingTransferSize* of the  
 14932 receiving device.

14933 **10.6.2.4.4.1 Payload Format**14934 **Figure 10-47. Format of the *TransferDataError* Command Payload**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	TunnelID (M)	TransferDataStatus (M)

14935 **10.6.2.4.4.2 Payload Details**

14936 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
 14937 triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or  
 14938 passed with any commands concerning that specific tunnel.

14939 **TransferDataStatus:** The TransferDataStatus parameter indicates the error that occurred within the receiver after  
 14940 the last TransferData command.

14941 The TransferDataStatus values are shown in Table 10-68.

14942 **Table 10-68. TransferDataStatus Values**

Value	Description	Remarks
0x00	No such tunnel	The <i>TransferData</i> command contains a <i>TunnelID</i> of a non-existent tunnel.
0x01	Wrong device	The <i>TransferData</i> command contains a <i>TunnelID</i> that does not match the device sending the data.
0x02	Data overflow	The <i>TransferData</i> command contains more data than indicated by the <i>MaximumIncomingTransferSize</i> of the receiving device

14943 **10.6.2.4.4.3 When Generated**

14944 Is generated if the server wants to tell the client that there was something wrong with the last TransferData  
 14945 command.

14946 **10.6.2.4.4.4 Effect on Receipt**

14947 Indicates that the client wants to tell the server that there was something wrong with the last TransferData  
 14948 command.

14949 **10.6.2.4.5 AckTransferData Command**

14950 Command sent in response to each TransferData command in case – and only in case – flow control has been  
 14951 requested by the client in the TunnelRequest command and is supported by both tunnel endpoints. The response  
 14952 payload indicates the number of octets that may still be received by the receiver.

14953 **10.6.2.4.5.1 Payload Format**

14954 **Figure 10-48. Format of the AckTransferData Command Payload**

<b>Octets</b>	2	2
<b>Data Type</b>	uint16	uint16
<b>Field Name</b>	TunnelID (M)	NumberOfBytesLeft (M)

14955 **10.6.2.4.5.2 Payload Details**

14956 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
 14957 triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or  
 14958 passed with any commands concerning that specific tunnel.

14959 **NumberOfBytesLeft:** Indicates the number of bytes that may still be received by the initiator of this command  
 14960 (receiver). It is most likely the remaining size of the buffer holding the data that is sent over TransferData. As an  
 14961 example: A value of 150 indicates that the next TransferData command must not contain more than 150 bytes of  
 14962 payload or data will get lost. A value of 0 indicates that there is no more space left in the receiver and the sender  
 14963 should completely stop sending data. After the reception of a ReadyData command, the sender may continue its  
 14964 data transfer.

14965 **10.6.2.4.5.3 When Generated**

14966 If flow control is on, the command is issued by the server to inform the client that the last TransferData command  
14967 has been successfully received and how much space is left to receive further data.

#### 14968 **10.6.2.4.5.4 Effect on Receipt**

14969 If flow control is on, the reception of this command indicates that the client wants to inform the server that the last  
14970 TransferData command has been successfully received and how much space is left to receive further data.

### 14971 **10.6.2.4.6 ReadyData Command**

14972 The ReadyData command is generated – after a receiver had to stop the dataflow using the AckTransferData(0)  
14973 command – to indicate that the device is now ready to continue receiving data. The parameter NumberOfOctetsLeft  
14974 gives a hint on how much space is left for the next data transfer. The ReadyData command is only issued if flow  
14975 control is enabled.

#### 14976 **10.6.2.4.6.1 Payload Format**

14977 **Figure 10-49. Format of the ReadyData Command Payload**

<b>Octets</b>	2	2
<b>Data Type</b>	uint16	uint16
<b>Field Name</b>	TunnelID (M)	NumberOfOctetsLeft (M)

#### 14978 **10.6.2.4.6.2 Payload Details**

14979 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
14980 triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or  
14981 passed with any commands concerning that specific tunnel.

14982 **NumberOfOctetsLeft:** Indicates the number of octets that may be received by the initiator of this command  
14983 (receiver). It is most likely the remaining size of the buffer holding the data that is sent over TransferData. As an  
14984 example: A value of 150 indicates that the next TransferData command must not contain more than 150 bytes of  
14985 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  
14986 the cluster to decide what flow control algorithm shall be applied.

#### 14987 **10.6.2.4.6.3 When Generated**

14988 If generated by the server, this command informs the client that it may now continue to send and how much space is  
14989 left within the server to receive further data.

#### 14990 **10.6.2.4.6.4 Effect on Receipt**

14991 If received by the server, this command informs the server that it may now continue to send and how much space is  
14992 left within the client to receive further data.

### 14993 **10.6.2.4.7 Get Supported Tunnel Protocols Command**

14994 Get Supported Tunnel Protocols is the client command used to determine the tunnel protocols supported on another  
14995 device.

#### 14996 **10.6.2.4.7.1 Payload Format**

14997

**Figure 10-50. Format of the *Get Supported Tunnel Protocols* Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	uint8
<b>Field Name</b>	Protocol Offset

14998 **10.6.2.4.7.2 Payload Details**

14999 **Protocol Offset:** Where there are more protocols supported than can be returned in a single Supported Tunnel  
 15000 Protocols Response command, this field allows an offset to be specified on subsequent Get Supported Tunnel  
 15001 Protocols commands. An offset of zero (0x00) should be used for an initial (or only) Get Supported Tunnel  
 15002 Protocols command (indicating that the returned list of protocols should commence with first available protocol). As  
 15003 a further example, if 10 protocols had previously been returned, the next Get Supported Tunnel Protocols  
 15004 command should use an offset of 10 (0x0A) to indicate the 11th available protocol should be the first returned  
 15005 in the next response.

15006 **10.6.2.4.7.3 Effect on Receipt**

15007 On receipt of this command, a device will respond with a Supported Tunnel Protocols Response command,  
 15008 indicating the tunnel protocols it supports (see sub- clause 10.6.2.5.6 for further details).

15009 **10.6.2.5 Commands Generated**

15010 Table 10-69 lists commands that are generated by the server.

15011 **Table 10-69. Cluster-Specific Commands Sent by the Server**

<b>Command Identifier Field Value</b>	<b>Description</b>	<b>M/O</b>
0x00	<i>RequestTunnelResponse</i>	M
0x01	<i>TransferData</i>	M
0x02	<i>TransferDataError</i>	M
0x03	<i>AckTransferData</i>	O
0x04	<i>ReadyData</i>	O
0x05	<i>Supported Tunnel Protocols Response</i>	O
0x06	<i>TunnelClosureNotification</i>	O

15012 **10.6.2.5.1 RequestTunnelResponse Command**

15013 RequestTunnelResponse is sent by the server in response to a RequestTunnel command previously received from  
 15014 the client. The response contains the status of the RequestTunnel command and a tunnel identifier corresponding to  
 15015 the tunnel that has been set-up in the server in case of success.

15016 **10.6.2.5.1.1 Payload Format**

15017

**Figure 10-51. Format of the *RequestTunnelResponse* Command Payload**

Octets	2	1	2
Data Type	uint16	uint8	uint16
Field Name	TunnelID (M)	TunnelStatus (M)	Maximum Incoming TransferSize

15018 **10.6.2.5.1.2 Payload Details**

15019 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
 15020 triggered through the RequestTunnel command. This ID must now be used to send data through this tunnel  
 15021 (TunnelID, TransferData) and is also required to close the tunnel again (CloseTunnel). If the command has failed,  
 15022 the TunnelStatus contains the reason of the error and the TunnelID is set to 0xFFFF.

15023 **TunnelStatus:** The TunnelStatus parameter indicates the server's internal status after the execution of a  
 15024 RequestTunnel command.

15025 The TunnelStatus values are shown in Table 10-70.

15026 **Table 10-70. TunnelStatus Values**

Value	Description	Remarks
0x00	Success	The tunnel has been opened and may now be used to transfer data in both directions.
0x01	Busy	The server is busy and cannot create a new tunnel at the moment. The client may try again after a recommended timeout of 3 minutes.
0x02	No more tunnel IDs	The server has no more resources to setup requested tunnel. Clients should close any open tunnels before retrying.
0x03	Protocol not supported	The server does not support the protocol that has been requested in the ProtocolID parameter of the <i>RequestTunnel</i> command.
0x04	Flow control not supported	Flow control has been requested by the client in the <i>RequestTunnel</i> command but cannot be provided by the server (missing resources or no support).

15027

15028 **MaximumIncomingTransferSize:** A value that defines the size, in octets, of the maximum data packet that can  
 15029 be transferred to the server in the payload of a single TransferData command.

15030 **10.6.2.5.1.3 When Generated**

15031 Is generated in reply to a RequestTunnel command to inform the client about the result of the request.

15032 **10.6.2.5.1.4 Effect on Receipt**

15033 Should never be received by the server.

15034 **10.6.2.5.2 TransferData Command**

15035 Command that transfers data from server to the client. The data itself has to be placed within the payload.

15036 **10.6.2.5.2.1 Payload Format**

15037

**Figure 10-52. Format of the *TransferData* Command Payload**

<b>Octets</b>	2	Variable
<b>Data Type</b>	uint16	opaque
<b>Field Name</b>	TunnelID (M)	Data (M)

15038 **10.6.2.5.2.2 Payload Details**

15039 **TunnelID:** A number between 0..65535 that uniquely identifies the tunnel that has been allocated in the server  
 15040 triggered through the RequestTunnel command. This ID must be used for the data transfer through the tunnel or  
 15041 passed with any commands concerning that specific tunnel.

15042 **Data:** Series of octets containing the data to be transferred through the tunnel in the format of the communication  
 15043 protocol for which the tunnel has been requested and opened. The payload containing the assembled data  
 15044 exactly as it has been sent away by the client. Theoretically, its length is solely limited through the fragmentation  
 15045 algorithm and the RX/TX transfer buffer sizes within the communication partners. The content of the payload is  
 15046 up to the application sending the data. It is not guaranteed that it contains a complete PDU, nor is any assumption  
 15047 to be made on its internal format (which is left up to the implementer of the specific tunnel protocol).

15048 **10.6.2.5.2.3 When Generated**

15049 Is generated when the server wants to tunnel protocol data to the client.

15050 **10.6.2.5.2.4 Effect on Receipt**

15051 Indicates that the server has received tunneled protocol data from the client.

15052 **10.6.2.5.3 TransferDataError Command**

15053 See sub-clause 10.6.2.4.4.

15054 **10.6.2.5.4 AckTransferData Command**

15055 See sub-clause 10.6.2.4.5.

15056 **10.6.2.5.5 ReadyData Command**

15057 See sub-clause 10.6.2.4.6.

15058 **10.6.2.5.6 Supported Tunnel Protocols Response Command**

15059 Supported Tunnel Protocols Response is sent in response to a Get Supported Tunnel Protocols command  
 15060 previously received. The response contains a list of tunnel protocols supported by the device; the payload of the  
 15061 response should be capable of holding up to 16 protocols.

15062 **10.6.2.5.6.1 Payload Format**

15063 **Figure 10-53. Format of the *Supported Tunnel Protocols Response* Command Payload**

<b>Octets</b>	1	1	3	...	3
<b>Data Type</b>	bool	uint8			
<b>Field Name</b>	Protocol List Complete	Protocol Count	Protocol 1	...	Protocol n

15064

15065 where each protocol field shall be formatted as:

15066 **Figure 10-54. Format of the Supported Tunnel Protocols Response Command Protocol Fields**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	enum8
<b>Field Name</b>	Manufacturer Code	Protocol ID

15067 **10.6.2.5.6.2 Payload Details**

15068 **Protocol List Complete:** The Protocol List Complete field is a Boolean; a value of 0 indicates that there are more  
15069 supported protocols available (if more than 16 protocols are supported). A value of 1 indicates that the list of  
15070 supported protocols is complete.

15071 **Protocol Count:** The number of Protocol fields contained in the response.

15072 **Manufacturer Code:** A code that is allocated by the ZigBee Alliance, relating the manufacturer to a device and - for  
15073 tunneling - a manufacturer specific protocol. A value of 0xFFFF indicates a standard (i.e. non- manufacturer  
15074 specific) protocol

15075 **Protocol ID:** An enumeration representing the identifier of the metering communication protocol for the supported  
15076 tunnel. Table 10-67 lists the possible values for standard protocols

15077 **10.6.2.5.6.3 When Generated**

15078 Is generated in reply to a Get Supported Tunnel Protocols command, to indicate the tunnel protocols supported by  
15079 the device

15080 **10.6.2.5.7 TunnelClosureNotification Command**

15081 TunnelClosureNotification is sent by the server to indicate that a tunnel has been closed due to expiration of a  
15082 CloseTunnelTimeout.

15083 **10.6.2.5.7.1 Payload Format**

15084 **Figure 10-55. Format of the TunnelClosureNotification Command Payload**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	TunnelID (M)

15085 **10.6.2.5.7.2 Payload Details**

15086 **TunnelID:** The identifier of the tunnel that has been closed. It is the same number that has been previously returned  
15087 in the response to a RequestTunnel command. Valid numbers range between 0..65535 and must correspond to a  
15088 tunnel that was still active and maintained by the server.

15089 **10.6.2.5.7.3 When Generated**

15090 The command is sent by a server when a tunnel is closed due to expiration of CloseTunnelTimeout. It is sent  
15091 unicast to the client that had originally requested that tunnel.



## 15092 **10.6.3 Client**

### 15093 **10.6.3.1 Dependencies**

15094 This cluster requires APS fragmentation [Z1] to be implemented, with maximum transfer sizes defined by the  
15095 device's negotiated input buffer sizes.

### 15096 **10.6.3.2 Attributes**

15097 The client has no cluster specific attributes.

### 15098 **10.6.3.3 Commands Received**

15099 The client receives the cluster-specific response commands detailed in 10.6.2.5.

### 15100 **10.6.3.4 Commands Generated**

15101 The client generates the cluster-specific commands detailed in 10.6.2.4, as required by the application.

## 15102 **10.7 Key Establishment**

---

### 15103 **10.7.1 Scope and Purpose**

15104 This section specifies a cluster that contains commands and attributes necessary for managing secure communication  
15105 between devices.

15106 This section should be used in conjunction with the ZigBee Cluster Library, Foundation Specification (see  
15107 Chapter 2), which gives an overview of the library and specifies the frame formats and general commands used  
15108 therein.

15109 This version is specifically for inclusion in the Smart Energy profile. The document which originates from [Z10]  
15110 will continue to be developed in a backward-compatible manner as a more general secure communication cluster for  
15111 ZigBee applications as a whole.

### 15112 **10.7.2 General Description**

#### 15113 **10.7.2.1 Introduction**

15114 As previously stated, this document describes a cluster for managing secure communication in ZigBee. The  
15115 cluster is for Key Establishment.

#### 15116 **10.7.2.2 Network Security**

15117 The Key Establishment Cluster has been designed to be used where the underlying network security cannot be  
15118 trusted. As such, no information that is confidential information will be transported.

#### 15119 **10.7.2.3 Key Establishment**

15120 To allow integrity and confidentiality of data passed between devices, cryptographic schemes need to be deployed.  
15121 The cryptographic scheme deployed in the ZigBee Specification for frame integrity and confidentiality is based

15122 upon a variant of the AES-CCM described in [N3] called AES-CCM\*. This relies on the existence of secret keying  
15123 material shared between the involved devices. There are methods to distribute this secret keying material in a trusted  
15124 manner. However, these methods are generally not scalable or communication may be required with a trusted key  
15125 allocation party over an insecure medium. This leads to the requirement for automated key establishment schemes to  
15126 overcome these problems.

15127 Key establishment schemes can be effected<sup>123</sup> using either a key agreement scheme or a key transport scheme. The  
15128 key establishment scheme described in this document uses a key agreement scheme, therefore key transport schemes  
15129 will not be considered further in this document.

15130 A key agreement scheme is where both parties contribute to the shared secret and therefore the secret keying  
15131 material to be established is not sent directly; rather, information is exchanged between both parties that allows each  
15132 party to derive the secret keying material. Key agreement schemes may use either symmetric key or asymmetric key  
15133 (public key) techniques. The party that begins a key agreement scheme is called the initiator, and the other party is  
15134 called the responder.

15135 Key establishment using key agreement involves an initiator and a responder and four steps:

- 15136 1. Establishment of a trust relationship
- 15137 2. Exchange of ephemeral data
- 15138 3. Use of this ephemeral data to derive secret keying material using key agreement
- 15139 4. Confirmation of the secret keying material.

15140 There are two basic types of key establishment that can be implemented:

- 15141 • Symmetric Key Key Establishment
- 15142 • Public Key Key Establishment

#### 15143 **10.7.2.4 Symmetric Key Key Establishment**

15144 Symmetric Key Key Establishment (SKKE) is based upon establishing a link key based on a shared secret (master  
15145 key). If the knowledge of the shared secret is compromised, the established link key can also be compromised. If the  
15146 master key is publicly known or is set to a default value, it is known as Unprotected Key Establishment (UKE).  
15147 SKKE is the key establishment method used in the ZigBee specification therefore it will not be considered any  
15148 further.

#### 15149 **10.7.2.5 Public Key Key Establishment**

15150 Public Key Key Establishment (PKKE) is based upon establishing a link key based on shared static and  
15151 ephemeral public keys. As the public keys do not require any secrecy, the established link key cannot be  
15152 compromised by knowledge of them.

15153 As a device's static public key is used as part of the link key creation, it can either be transported independently to  
15154 the device's identity where binding between the two is assumed, or it can be transported as part of a implicit  
15155 certificate signed by a Certificate Authority, which provides authentication of the binding between the device's  
15156 identity and its public key as part of the key establishment process. This is called Certificate-Based Key  
15157 Establishment (CBKE) and is discussed in more detail in sub-clause 10.7.4.2.

15158 CBKE provides the most comprehensive form of Key Establishment and therefore will be the method specified in this  
15159 cluster.

15160 The purpose of the key agreement scheme as described in this document is to produce shared secret keying  
15161 material which can be subsequently used by devices using AES-CCM\* the cryptographic scheme deployed in the  
15162 ZigBee Specification or for any proprietary security mechanism implemented by the application.

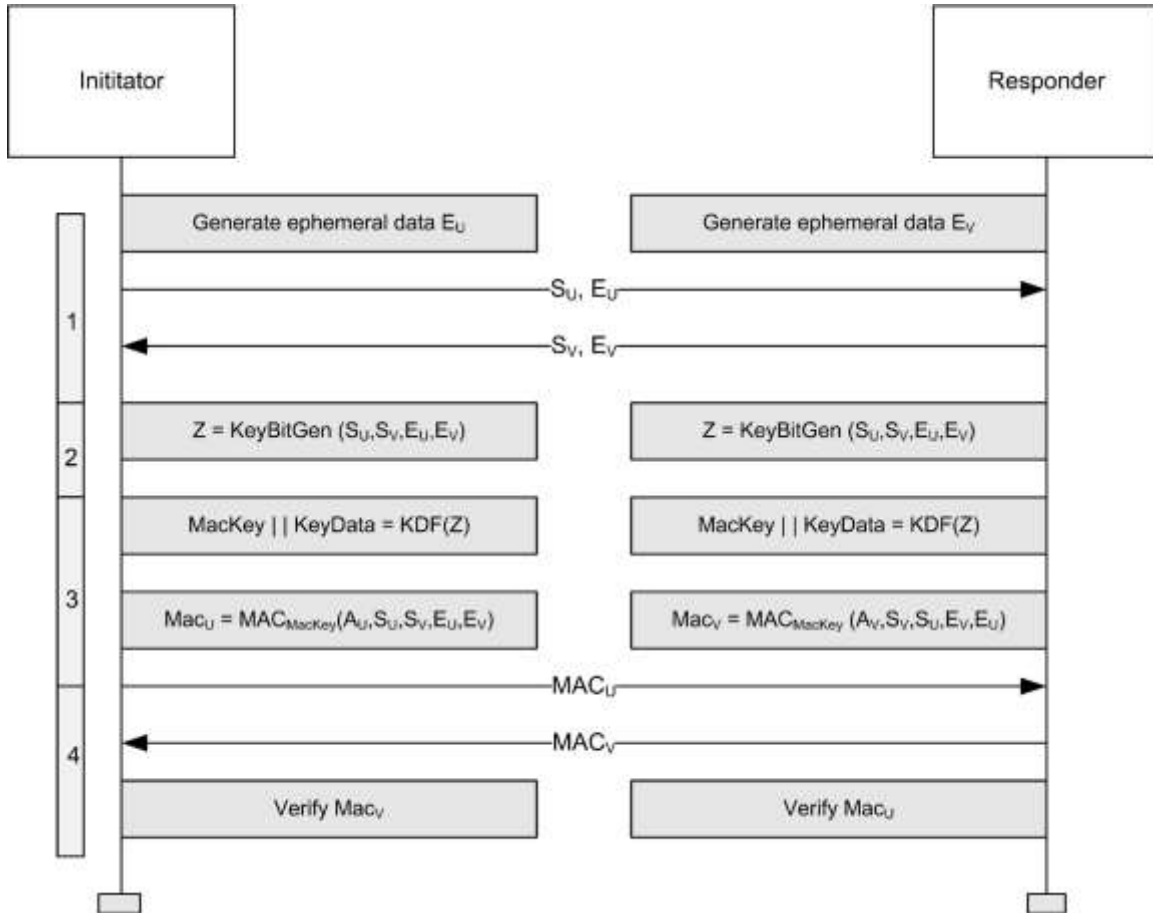
---

<sup>123</sup> CCB 2288

15163 **10.7.2.6 General Exchange**

15164 Figure 10-56 shows an overview of the general exchange which takes place between initiator and responder to  
 15165 perform key establishment.

15166 **Figure 10-56. Overview of General Exchange**



15167

15168

15169 The functions are:

- 15170 1. Exchange Static and Ephemeral Data
- 15171 2. Generate Key Bitstream
- 15172 3. Derive MAC key and Key Data
- 15173 4. Confirm Key using MAC

15174 The functions shown in Figure 10-56 depend on the Key Establishment mechanism.

15175 **10.7.2.6.1 Exchange Static and Ephemeral Data**

15176 Figure 10-56 shows static data  $S_U$  and  $S_V$ . For PKKE schemes, this represents a combination of the 64-bit device  
 15177 address [Z11] and the device’s static public key. The identities are needed by the MAC scheme and the static public  
 15178 keys are needed by the key agreement scheme.

15179 Figure 10-56 also shows ephemeral data  $E_U$  and  $E_V$ . For PKKE schemes, this represents the public key of a  
 15180 randomly generated key pair.

15181 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.

#### 15182 **10.7.2.6.2 Generate Key Bitstream**

15183 Figure 10-56 shows the KeyBitGen function for generating the key bitstream. The function's four parameters are  
15184 the identifiers and the ephemeral data for both devices. This ensures the same key is generated at both ends.

15185 For PKKE schemes, this is the ECMQV key agreement schemes specified in Section 6.2 of SEC1 [O1]. The  
15186 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  
15187  $Q_{1,V}$  of party V, the ephemeral data  $E_U$  represents the ephemeral public key  $Q_{2,U}$  of party U and the ephemeral data  
15188  $E_V$  represents the ephemeral public key  $Q_{2,V}$  of party V.

#### 15189 **10.7.2.6.3 Derive MAC Key and Key Data**

15190 Figure 10-56 shows the KDF (KeyDerivation Function) for generating the MAC Key and key data. The MAC  
15191 Key is used with a keyed hash message authentication function to generate a MAC and the key data is the shared  
15192 secret, e.g. the link key itself required for frame protection.

15193 For PKKE schemes, this is the key derivation function as specified in Section 3.6.1 of SEC1 [O1]. Note there is  
15194 no SharedInfo parameter of the referenced KDF, i.e. it is a null octet string of length 0.

15195 Figure 10-56 also shows generation of the MAC using the MAC Key derived using the KDF using a message  
15196 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  
15197 different for initiator and responder.

15198 For PKKE schemes, this is the MAC scheme specified in section 3.7 of SEC1 [O1]. The MAC in the reference is  
15199 the keyed hash function for message authentication specified in sub-clause 10.7.4.2.2.6 and the message M is a  
15200 concatenation of the identity (the 64-bit device address [E1]) of U, the identity of V and point-compressed octet-  
15201 string representations of the ephemeral public keys of parties U and V. The order of concatenation depends on  
15202 whether it is the initiator or responder. The additional component A is the single octet  $02_{16}$  for the initiator and  
15203  $03_{16}$  for the responder.

#### 15204 **10.7.2.6.4 Confirm Key Using MAC**

15205 Figure 10-56 shows MACs  $MAC_U$  and  $MAC_V$

15206 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  
15207 and compare it with the data received.

### 15208 **10.7.3 Cluster List**

15209 The clusters specified in this document are listed in Table 10-71.

15210 For our purposes, any device that implements the client side of this cluster may be considered the initiator of the  
15211 secure communication transaction.

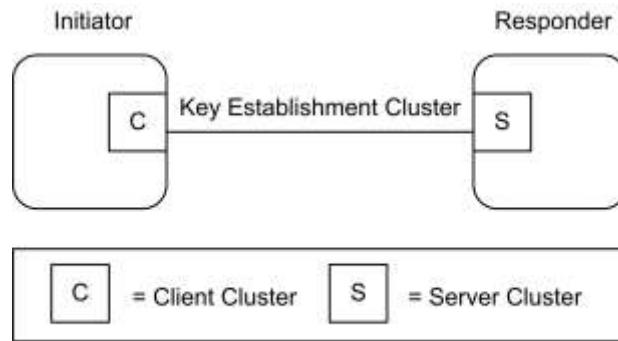
15212 **Table 10-71. Clusters Specified for the Secure Communication Functional Domain**

Cluster Name	Description
Key Establishment	Attributes and commands for establishing a shared secret between two ZigBee devices

15213

15214

**Figure 10-57. Typical Usage of the Key Establishment Cluster**



15215

*Note: Device names are examples for illustration purposes only*

## 15216 10.7.3.1 Key Establishment Cluster

### 15217 10.7.3.1.1 Overview

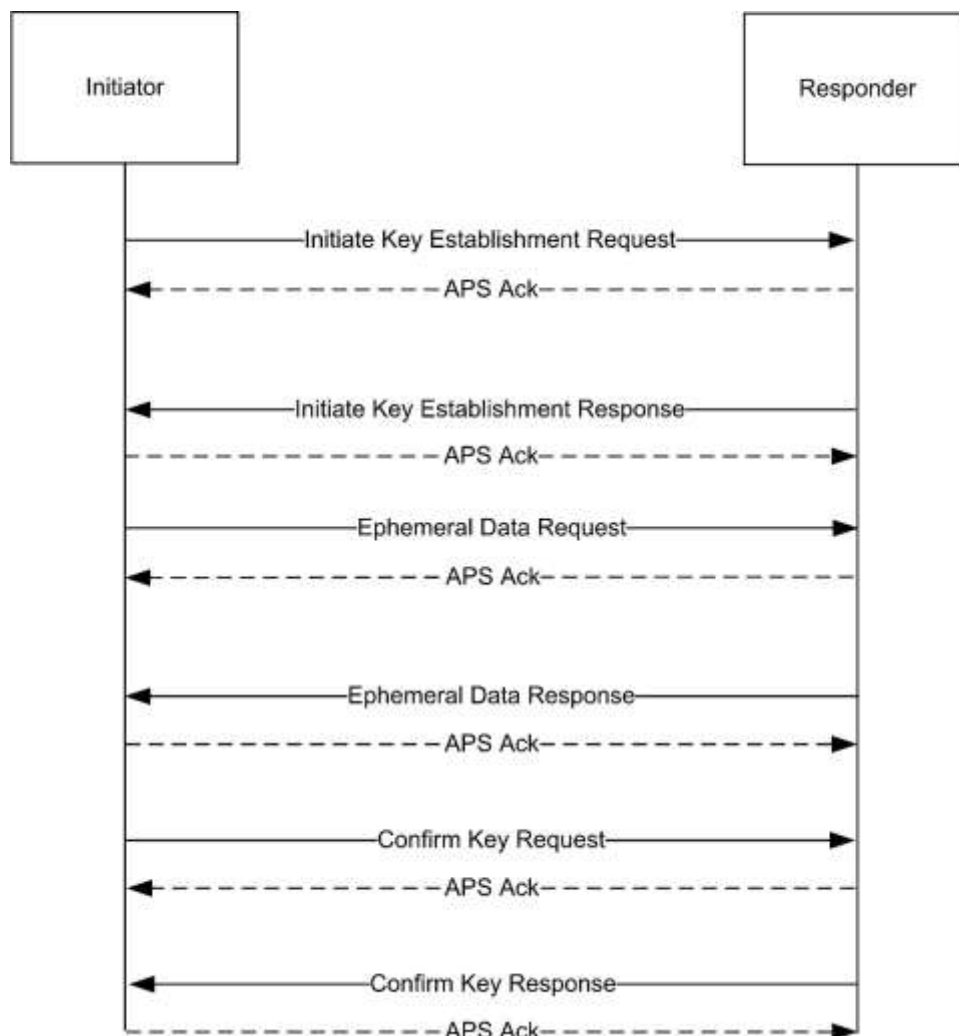
15218 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
15219 etc.

15220 This cluster provides attributes and commands to perform mutual authentication and establish keys between two  
15221 ZigBee devices. Figure 10-58 depicts a diagram of a successful key establishment negotiation.

15222

15223

**Figure 10-58. Key Establishment Command Exchange**



15224

15225

15226 As depicted above, all Key Establishment messages should be sent with APS retries enabled. A failure to  
 15227 receive an ACK in a timely manner can be seen as a failure of key establishment. No Terminate Key  
 15228 Establishment should be sent to the partner of device that has timed out the operation.

15229 The initiator can initiate the key establishment with any active endpoint on the responder device that supports the  
 15230 key establishment cluster. The endpoint can be either preconfigured or discovered, for example, by using ZDO  
 15231 Match-Desc-req. A link key successfully established using key establishment is valid for all endpoints on a  
 15232 particular device. The responder shall respond to the initiator using the source endpoint of the initiator's messages as  
 15233 the destination endpoint of the responder's messages.

15234 It is expected that the time it takes to perform the various cryptographic computations of the key establishment  
 15235 cluster may vary greatly based on the device. Therefore rather than set static timeouts, the Initiate Key  
 15236 Establishment Request and Response messages will contain approximate values for how long the device will take to  
 15237 generate the ephemeral data and how long the device will take to generate confirm key message.

15238 A device performing key establishment can use this information in order to choose a reasonable timeout for  
 15239 its partner during those operations. The timeout should also take into consideration the time it takes for a message  
 15240 to traverse the network including APS retries. A minimum transmission time of 2 seconds is recommended.

15241 For the Initiate Key Establishment Response message, it is recommended the initiator wait at least 2 seconds  
 15242 before timing out the operation. It is not expected that generating an Initiate Key Establishment Response will take  
 15243 significant time compared to generating the Ephemeral Data and Confirm Key messages.

15244 **10.7.3.1.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

15245 **10.7.3.1.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Application	SEKE

15246 **10.7.3.1.1.3 Cluster Identifiers**

Identifier	Name
0x0800	Key Establishment (Smart Energy)

15247 **10.7.3.1.2 Server**

15248 **10.7.3.1.2.1 Dependencies**

15249 The Key Establishment server cluster has no dependencies.

15250 **10.7.3.1.2.2 Attributes**

15251 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 15252 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify  
 15253 the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute  
 15254 sets are listed in Table 10-72.

15255 **Table 10-72. Key Establishment Attribute Sets**

Attribute Set Identifier	Description
0x000	Information

15256 **10.7.3.1.2.2.1 Information**

15257 The *Information* attribute set contains the attributes summarized in Table 10-73.

15258 **Table 10-73. Information Attribute Sets**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>KeyEstablishmentSuite</i>	enum16	0x0000 - 0xFFFF	R	0x0000	M

15259 **10.7.3.1.2.2.1.1 KeyEstablishmentSuite Attribute**

15260 The *KeyEstablishmentSuite* attribute is 16-bits in length and specifies all the cryptographic schemes for key  
 15261 establishment on the device. A device shall set the corresponding bit to 1 for every cryptographic scheme that is  
 15262 supports. All other cryptographic schemes and reserved bits shall be set to 0.

15263

**Table 10-74. Values of the *KeyEstablishmentSuite* Attribute**

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

15264 **10.7.3.1.2.3 Commands Received**

15265 The server side of the key establishment cluster is capable of receiving the commands listed in Table 10-75.

15266

**Table 10-75. Received Command IDs for the Key Establishment Cluster Server**

Command Identifier Field	Description	M/O
0x00	<i>Initiate Key EstablishmentRequest</i>	M
0x01	<i>Ephemeral Data Request</i>	M
0x02	<i>Confirm Key Data Request</i>	M
0x03	<i>Terminate Key Establishment</i>	M

15267 **10.7.3.1.2.3.1 Initiate Key Establishment Request Command**

15268 The Initiate Key Establishment Request command allows a device to initiate key establishment with another device.

15269 The sender will transmit its identity information and key establishment protocol information to the receiving device.

15270 **10.7.3.1.2.3.1.1 Payload Format**

15271 The Initiate Key Establishment Request command payload shall be formatted as illustrated in Figure 10-59.

15272

**Figure 10-59. Format of the *Initiate Key Establishment Request* Command Payload**

Octets	2	1	1	48
Data Type	map16	uint8	uint8	opaque
Field Name	Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

15273

15274 **Key Establishment Suite:** This will be the type of Key Establishment that the initiator is requesting for the Key  
15275 Establishment Cluster. For CBKE-ECMQV this will be 0x0001.15276 **Ephemeral Data Generate Time:** This value indicates approximately how long the initiator device will take in  
15277 seconds to generate the Ephemeral Data Request command. The valid range is 0x00 to 0xFE.15278 **Confirm Key Generate Time:** This value indicates approximately how long the initiator device will take in  
15279 seconds to generate the Confirm Key Request command. The valid range is 0x00 to 0xFE.15280 **Identity field:** For KeyEstablishmentSuite = 0x0001 (CBKE), the identity field shall be the block of octets  
15281 containing the implicit certificate CERTU as specified in sub-clause 10.7.4.2.15282 **10.7.3.1.2.3.1.2 Effect on Receipt**15283 If the device does not currently have the resources to respond to a key establishment request it shall send a  
15284 Terminate Key Establishment command with the result value set to NO\_RESOURCES and the Wait Time field shall  
15285 be set to an approximation of the time that must pass before the device will have the resources to process a new Key  
15286 Establishment Request.



15287 If the device can process this request, it shall check the Issuer field of the device's implicit certificate. If the Issuer  
 15288 field does not contain a value that corresponds to a known Certificate Authority, the device shall send a Terminate  
 15289 Key Establishment command with the result set to UNKNOWN\_ISSUER.

15290 If the device accepts the request it shall send an Initiate Key Establishment Response command containing  
 15291 its own identity information. The device should verify the certificate belongs to the address that the device is  
 15292 communicating with. The binding between the identity of the communicating device and its address is verifiable  
 15293 using out-of-band method.

15294 **10.7.3.1.2.3.2 Ephemeral Data Request Command**

15295 The Ephemeral Data Request command allows a device to communicate its ephemeral data to another device  
 15296 and request that the device send back its own ephemeral data.

15297 10.7.3.1.2.3.2.1 Payload Format

15298 **Figure 10-60. Format of the Ephemeral Data Request Command Payload**

<b>Octets</b>	22
<b>Data Type</b>	opaque
<b>Field Name</b>	Ephemeral Data (QEU)

15299 10.7.3.1.2.3.2.2 Effect on Receipt

15300 If the device is not currently in the middle of negotiating Key Establishment with the sending device when it  
 15301 receives this message, it shall send back a Terminate Key Establishment message with a result of  
 15302 BAD\_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this  
 15303 message in response to an Initiate Key Establishment Response command, it shall send back a Terminate Key  
 15304 Establishment message with a result of BAD\_MESSAGE. If the device can process the request it shall respond  
 15305 by generating its own ephemeral data and sending an Ephemeral Data Response command containing that value.

15306 **10.7.3.1.2.3.3 Confirm Key Request Command**

15307 The Confirm Key Request command allows the initiator sending device to confirm the key established with the  
 15308 responder receiving device based on performing a cryptographic hash using part of the generated keying material  
 15309 and the identities and ephemeral data of both parties.

15310 10.7.3.1.2.3.3.1 Payload Format

15311 The Confirm KeyRequest command payload shall be formatted as illustrated in Figure 10-61.

15312 **Figure 10-61. Format of the Confirm Key Request Command Payload**

<b>Octets</b>	16
<b>Data Type</b>	opaque
<b>Field Name</b>	Secure Message Authentication Code (MACU)

15313  
 15314 **Secure Message Authentication Code field:** The Secure Message Authentication Code field shall be the octet  
 15315 representation of MACU as specified in sub-clause 10.7.4.2.

15316 10.7.3.1.2.3.3.2 Effect on Receipt

15317 If the device is not currently in the middle of negotiating Key Establishment with the sending device when it  
 15318 receives this message, it shall send back a Terminate Key Establishment message with a result of  
 15319 BAD\_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this

15320 message in response to an Ephemeral Data Response command, it shall send back a Terminate Key  
15321 Establishment message with a result of BAD\_MESSAGE.

15322 On receipt of the Confirm Key Request command the responder device shall compare the received MACU  
15323 value with its own reconstructed version of MACU. If the two match the responder shall send back MACV by  
15324 generating an appropriate Confirm Key Response command. If the two do not match, the responder shall send  
15325 back a Terminate Key Establishment with a result of BAD\_KEY\_CONFIRM and terminate the key establishment.

#### 15326 10.7.3.1.2.3.4 Terminate Key Establishment Command

15327 The Terminate Key Establishment command may be sent by either the initiator or responder to indicate a failure in  
15328 the key establishment exchange.

##### 15329 10.7.3.1.2.3.4.1 Payload Format

15330 The Terminate Key Establishment command payload shall be formatted as illustrated in Figure 10-62.

15331 **Figure 10-62. Format of the Terminate Key Establishment Command Payload**

<b>Octets</b>	1	1	2
<b>Data Type</b>	enum8	uint8	map16
<b>Field Name</b>	Status Code	Wait Time	KeyEstablishmentSuite

15332

15333 **Status Field:** The Status field shall be one of the error codes in Table 10-76.

15334 **Table 10-76. Terminate Key Establishment Command Status Field**

Enumeration	Value	Description
UNKNOWN_ISSUER	0x01	The Issuer field within the key establishment partner's certificate is unknown to the sending device, and it has terminated the key establishment.
BAD_KEY_CONFIRM	0x02	The device could not confirm that it shares the same key with the corresponding device and has terminated the key establishment.
BAD_MESSAGE	0x03	The device received a bad message from the corresponding device (e.g. message with bad data, an out of sequence number, or a message with a bad format) and has terminated the key establishment.
NO_RESOURCES	0x04	The device does not currently have the internal resources necessary to perform key establishment and has terminated the exchange.
UNSUPPORTED_SUITE	0x05	The device does not support the specified key establishment suite in the partner's Initiate Key Establishment message.

15335

15336 **Wait Time:** This value indicates the minimum amount of time in seconds the initiator device should wait before  
15337 trying to initiate key establishment again. The valid range is 0x00 to 0xFE.

15338 **KeyEstablishmentSuite:** This value will be set the value of the KeyEstablishmentSuite attribute. It indicates the list  
15339 of key exchange methods that the device supports.

##### 15340 10.7.3.1.2.3.4.2 Effect on Receipt

15341 On receipt of the Terminate Key Establishment command the device shall terminate key establishment with the  
15342 sender. If the device receives a status of BAD\_MESSAGE or NO\_RESOURCES it shall wait at least the time  
15343 specified in the Wait Time field before trying to re-initiate Key Establishment with the device.

15344 If the device receives a status of UNKNOWN\_SUITE it should examine the KeyEstablishmentSuite field to  
 15345 determine if another suite can be used that is supported by the partner device. It may re-initiate key establishment  
 15346 using that one of the supported suites after waiting the amount of time specified in the Wait Time field. If the device  
 15347 does not support any of the types in the KeyEstablishmentSuite field, it should not attempt key establishment again  
 15348 with that device.

15349 If the device receives a status of UNKNOWN\_ISSUER or BAD\_KEY\_CONFIRM the device should not attempt  
 15350 key establishment again with the device, as it is unlikely that another attempt will be successful.

15351 **10.7.3.1.2.4 Commands Generated**

15352 The server generates the commands detailed in sub-clause 10.7.3.1.3.3, as well as those used for reading and  
 15353 writing attributes.

15354 **10.7.3.1.3 Client**

15355 **10.7.3.1.3.1 Dependencies**

15356 The Key Establishment client cluster has no dependencies.

15357 **10.7.3.1.3.2 Attributes**

15358 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 15359 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify  
 15360 the attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute  
 15361 sets are listed in Table 10-77.

15362 **Table 10-77. Key Establishment Attribute Sets**

Attribute Set Identifier	Description
0x000	Information

15363 **10.7.3.1.3.2.1 Information**

15364 The Information attribute set contains the attributes summarized in Table 10-78.

15365 **Table 10-78. Attributes of the Information Attribute Set**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>KeyEstablishmentSuite</i>	enum16	0x0000 – 0xFFFF	R	0x0000	M

15366 **10.7.3.1.3.2.1.1 KeyEstablishmentSuite Attribute**

15367 The KeyEstablishmentSuite attribute is 16-bits in length and specifies all the cryptographic schemes for key  
 15368 establishment on the device. A device shall set the corresponding bit to 1 for every cryptographic scheme that is  
 15369 supports. All other cryptographic schemes and reserved bits shall be set to 0. This attribute shall be set to one of  
 15370 the non-reserved values listed in Table 10-79.

15371 **Table 10-79. Values of the KeyEstablishmentSuite Attribute**

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

15372 **10.7.3.1.3.3 Commands Received**

15373 The client side of the Key Establishment cluster is capable of receiving the commands listed in Table 10-80.

15374 **Table 10-80. Received Command IDs for the Key Establishment Cluster Client**

Command Identifier Field Value	Description	M/O
0x00	<i>Initiate Key Establishment Response</i>	M
0x01	<i>Ephemeral Data Response</i>	M
0x02	<i>Confirm Key Data Response</i>	M
0x03	<i>Terminate Key Establishment</i>	M

### 15375 10.7.3.1.3.3.1 Initiate Key Establishment Response Command

15376 The Initiate Key Establishment Response command allows a device to respond to a device requesting the initiation  
15377 of key establishment with it. The sender will transmit its identity information and key establishment protocol  
15378 information to the receiving device.

#### 15379 10.7.3.1.3.3.1.1 Payload Format

15380 The Initiate Key Establishment Response command payload shall be formatted as illustrated in Figure 10-63.

15381 **Figure 10-63. Format of the Initiate Key Establishment Response Command Payload**

<b>Octets</b>	2	1	1	48
<b>Data Type</b>	map16	uint8	uint8	opaque
<b>Field Name</b>	Requested Key Establishment suite	Ephemeral Data Generate Time	Confirm Key Generate Time	Identity (IDU)

15382

15383 **Requested Key Establishment Suite:** This will be the type of KeyEstablishmentSuite that the initiator has  
15384 requested be used for the key establishment exchange. The device shall set a single bit in the bitmask indicating the  
15385 requested suite, all other bits shall be set to zero.

15386 **Ephemeral Data Generate Time:** This value indicates approximately how long in seconds the responder device  
15387 takes to generate the Ephemeral Data Response message. The valid range is 0x00 to 0xFE.

15388 **Confirm Key Generate Time:** This value indicates approximately how long the responder device will take in  
15389 seconds to generate the Confirm Key Response message. The valid range is 0x00 to 0xFE.

15390 **Identity field:** For KeyEstablishmentSuite = 0x0001 (CBKE), the identity field shall be the block of Octets  
15391 containing the implicit certificate CERTU as specified in sub-clause 10.7.4.2.

#### 15392 10.7.3.1.3.3.1.2 Effect on Receipt

15393 If the device is not currently in the middle of negotiating Key Establishment with the sending device when it  
15394 receives this message, it shall send back a Terminate Key Establishment message with a result of  
15395 BAD\_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this  
15396 message in response to an Initiate Key Establishment Request command, it shall send back a Terminate Key  
15397 Establishment message with a result of BAD\_MESSAGE.

15398 On receipt of this command the device shall check the Issuer field of the device's implicit certificate. If the Issuer  
15399 field does not contain a value that corresponds to a known Certificate Authority, the device shall send a Terminate  
15400 Key Establishment command with the status value set to UNKNOWN\_ISSUER. If the device does not currently  
15401 have the resources to respond to a key establishment request it shall send a Terminate Key Establishment  
15402 command with the status value set to NO\_RESOURCES and the Wait Time field shall be set to an approximation  
15403 of the time that must pass before the device has the resources to process the request.

15404 If the device accepts the response it shall send an Ephemeral Data Request command. The device should  
 15405 verify the certificate belongs to the address that the device is communicating with. The binding between the identity  
 15406 of the communicating device and its address is verifiable using out-of-band method.

15407 **10.7.3.1.3.3.2 Ephemeral Data Response Command**

15408 The Ephemeral Data Response command allows a device to communicate its ephemeral data to another device  
 15409 that previously requested it.

15410 10.7.3.1.3.3.2.1 Payload Format

15411 **Figure 10-64. Format of the *Ephemeral Data Response Command* Payload**

<b>Octets</b>	22
<b>Data Type</b>	opaque
<b>Field Name</b>	Ephemeral Data (QEV)

15412 10.7.3.1.3.3.2.2 Effect on Receipt

15413 If the device is not currently in the middle of negotiating Key Establishment with the sending device when it receives  
 15414 this message, it shall send back a Terminate Key Establishment message with a result of BAD\_MESSAGE. If the  
 15415 device is in the middle of Key Establishment with the sender but did not receive this message in response to an Ephemeral  
 15416 Data Request command, it shall send back a Terminate Key Establishment message with a result of BAD\_MESSAGE.

15417 On receipt of this command if the device can handle the request it shall perform key generation, key derivation, and  
 15418 MAC generation. If successful it shall generate an appropriate Confirm Key Request command, otherwise it shall  
 15419 generate a Terminate Key Establishment with a result value of NO\_RESOURCES.

15420 **10.7.3.1.3.3.3 Confirm Key Response Command**

15421 The Confirm Key Response command allows the responder to verify the initiator has derived the same secret key.  
 15422 This is done by sending the initiator a cryptographic hash generated using the keying material and the identities  
 15423 and ephemeral data of both parties.

15424 10.7.3.1.3.3.3.1 Payload Format

15425 The Confirm Key Response command payload shall be formatted as illustrated in Figure 10-65.

15426 **Figure 10-65. Format of the *Confirm Key Response Command* Payload**

<b>Octets</b>	16
<b>Data Type</b>	opaque
<b>Field Name</b>	Secure Message Authentication Code (MACV)

15427  
 15428 **Secure Message Authentication Code field:** The Secure Message Authentication Code field shall be the octet  
 15429 representation of MACV as specified in sub-clause 10.7.4.2.

15430 10.7.3.1.3.3.3.2 Effect on Receipt

15431 If the device is not currently in the middle of negotiating Key Establishment with the sending device when it  
 15432 receives this message, it shall send back a Terminate Key Establishment message with a result of  
 15433 BAD\_MESSAGE. If the device is in the middle of Key Establishment with the sender but did not receive this  
 15434 message in response to a Confirm Key Request command, it shall send back a Terminate Key Establishment  
 15435 message with a result of BAD\_MESSAGE.

15436 On receipt of the Confirm Key Response command the initiator device shall compare the received MACV  
 15437 value with its own reconstructed version of the MACV. If the two match then the initiator can consider the key  
 15438 establishment process to be successful. If the two do not match, the initiator should send a Terminate Key  
 15439 Establishment command with a result of BAD\_KEY\_CONFIRM.

#### 15440 10.7.3.1.3.3.4 Terminate Key Establishment Command

15441 The Terminate Key Establishment command may be sent by either the initiator or responder to indicate a failure in  
 15442 the key establishment exchange.

##### 15443 10.7.3.1.3.3.4.1 Payload Format

15444 **Figure 10-66. Format of the Terminate Key Establishment Command Payload**

Octets	1	1	2
Data Type	enum8	uint8	map16
Field Name	Status Code	Wait Time	KeyEstablishmentSuite

15445

15446 **Status field:** The Status field shall be one of the error codes shown in Table 10-81.

15447 **Table 10-81. Terminate Key Establishment Command Status Field**

Enumeration	Value	Description
UNKNOWN_ISSUER	0x01	The Issuer field within the key establishment partner's certificate is unknown to the sending device, and it has terminated the key establishment.
BAD_KEY_CONFIRM	0x02	The device could not confirm that it shares the same key with the corresponding device and has terminated the key establishment.
BAD_MESSAGE	0x03	The device received a bad message from the corresponding device (e.g. message with bad data, an out of sequence number, or a message with a bad format) and has terminated the key establishment.
NO_RESOURCES	0x04	The device does not currently have the internal resources necessary to perform key establishment and has terminated the exchange.
UNSUPPORTED_SUITE	0x05	The device does not support the specified key establishment suite in the partner's Initiate Key Establishment message.

15448

15449 **Wait Time:** This value indicates the minimum amount of time in seconds the initiator device should wait before  
 15450 trying to initiate key establishment again. The valid range is 0x00 to 0xFE.

15451 **KeyEstablishmentSuite:** This value will be set the value of the KeyEstablishmentSuite attribute. It indicates the list  
 15452 of key exchange methods that the device supports.

##### 15453 10.7.3.1.3.3.4.2 Effect on Receipt

15454 On receipt of the Terminate Key Establishment command the device shall terminate key establishment with the  
 15455 sender. If the device receives a status of BAD\_MESSAGE or NO\_RESOURCES it shall wait at least the time  
 15456 specified in the Wait Time field before trying to re-initiate Key Establishment with the device.

15457 If the device receives a status of UNKNOWN\_SUITE it should examine the KeyEstablishmentSuite field to  
15458 determine if another suite can be used that is supported by the partner device. It may re-initiate key  
15459 establishment using that one of the supported suites after waiting the amount of time specified in the Wait Time  
15460 field. If the device does not support any of the types in the KeyEstablishmentSuite field, it should not attempt key  
15461 establishment again with that device.

15462 If the device receives a status of UNKNOWN\_ISSUER or BAD\_KEY\_CONFIRM the device should not attempt  
15463 key establishment again with the device, as it is unlikely that another attempt will be successful.

#### 15464 **10.7.3.1.3.4 Commands Generated**

15465 The client generates the commands detailed in sub-clause 10.7.3.1.2.3, as well as those used for reading and  
15466 writing attributes.

### 15467 **10.7.4 Application Implementation**

#### 15468 **10.7.4.1 Network Security for Smart Energy Networks**

15469 The underlying network security for Smart Energy networks is assumed to be ZigBee Standard security using  
15470 pre-configured link keys.

15471 A temporary link key for a joining device is produced by performing the cryptographic hash function on a random  
15472 number assigned to the joining device (e.g. serial number) and the device identifier, which is the device's 64-bit  
15473 IEEE address [Z11].

15474 The joining device's assigned random number is then conveyed to the utility via an out-of-band mechanism (e.g.  
15475 telephone call, or web site registration). The utility then commissions the energy service interface (ESI) at the  
15476 premises where the joining device is by installing the temporary link key at the ESI on the back channel.

15477 When the joining device powers up, it will also create a temporary link key as above and therefore at the time of  
15478 joining both the joining device and the ESI have the same temporary link key, which can be used to transport the  
15479 network key securely to the joining device.

15480 At this point, the device will be considered joined and authenticated as far as network security is concerned.  
15481 The secure communication cluster can now be invoked to replace the temporary link key with a more secure link  
15482 key based on public key cryptography.

#### 15483 **10.7.4.2 Certificate-Based Key Establishment**

15484 The Certificate-Based Key-Establishment (CBKE) solution uses public-key technology with digital certificates and  
15485 root keys. Each device has a private key and a digital certificate that is signed by a Certificate Authority (CA).

15486 The digital certificate includes:

- 15487 • Reconstruction data for the device's public key
- 15488 • The device's extended 64-bit IEEE address
- 15489 • Profile specific information (e.g., the device class, network id, object type, validity date, etc.)

15490 Certificates provide a mechanism for cryptographically binding a public key to a device's identity and  
15491 characteristics.

15492 Trust for a CBKE solution is established by provisioning a CA root key and a digital certificate to each device.  
15493 A CA root key is the public key paired with the CA's private key. A CA uses its private key to sign digital  
15494 certificates and the CA root key is used to verify these signatures. The trustworthiness of a public key is confirmed  
15495 by verifying the CA's signature of the digital certificate. Certificates can be issued either by the device  
15496 manufacturer, the device distributor, or the end customer. For example, in practical situations, the CA may be a  
15497 computer (with appropriate key management software) that is kept physically secure at the end customer's facility  
15498 or by a third-party.

- 15499 At the end of successful completion of the CBKE protocol the following security services are offered:
- 15500 • Both devices share a secret link key.
  - 15501 • Implicit Key Authentication: Both devices know with whom they share this link key.
  - 15502 • Key Confirmation: Each device knows that the other device actually has computed the key correctly.
  - 15503 • No Unilateral Key Control: No device has complete control over the shared link key that is established.
  - 15504 • Perfect Forward Secrecy: If the private key gets compromised none of future and past communications are exposed.
  - 15505
  - 15506 • Known Key Security resilience: Each shared link key created per session is unique.

## 15507 **10.7.4.2.1 Notation and Representation**

### 15508 **10.7.4.2.1.1 Strings and String Operations**

15509 A string is a sequence of symbols over a specific set (e.g., the binary alphabet {0,1} or the set of all octets).  
15510 The length of a string is the number of symbols it contains (over the same alphabet). The right-concatenation of  
15511 two strings  $x$  and  $y$  of length  $m$  and  $n$  respectively (notation:  $x \parallel y$ ), is the string  $z$  of length  $m+n$  that coincides with  
15512  $x$  on its leftmost  $m$  symbols and with  $y$  on its rightmost  $n$  symbols. An octet is a bit string of length 8.

### 15513 **10.7.4.2.1.2 Integers and Their Representation**

15514 Throughout this specification, the representation of integers as bit strings or octet strings shall be fixed. All integers  
15515 shall be represented as binary strings in most-significant-bit first order and as octet strings in most-significant-  
15516 octet first order. This representation conforms to the convention in Section 2.3 of SEC1 [O1].

### 15517 **10.7.4.2.1.3 Entities**

15518 Throughout this specification, each entity shall be a DEV and shall be uniquely identified by its 64-bit IEEE  
15519 device address [Z11]. The parameter `entlen` shall have the integer value 64.

## 15520 **10.7.4.2.2 Cryptographic Building Blocks**

15521 The following cryptographic primitives and data elements are defined for use with the CBKE protocol specified in  
15522 this document.

### 15523 **10.7.4.2.2.1 Elliptic-Curve Domain Parameters**

15524 The elliptic curve domain parameters used in this specification shall be those for the curve “ansit163k1” as  
15525 specified in section 3.4.1 of SEC2 [O2].

15526 All elliptic-curve points (and operations here on) used in this specification shall be (performed) on this curve.

### 15527 **10.7.4.2.2.2 Elliptic-Curve Point Representation**

15528 All elliptic-curve points shall be represented as point compressed octet strings as specified in sections 2.3.3 and  
15529 2.3.4 of SEC1 [O1]. Thus, each elliptic-curve point can be represented in 22 bytes.

### 15530 **10.7.4.2.2.3 Elliptic-Curve Key Pair**

15531 An elliptic-curve-key pair consists of an integer  $d$  and a point  $Q$  on the curve determined by multiplying the  
15532 generating point  $G$  of the curve by this integer (i.e.,  $Q=dG$ ) as specified in section 3.2.1 of SEC1 [O1]. Here,  $Q$  is  
15533 called the public key, whereas  $d$  is called the private key; the pair  $(d, Q)$  is called the key pair. Each private key shall  
15534 be represented as specified in section 2.3.7 of SEC1 [O1]. Each public key shall be represented as defined in sub-  
15535 clause 10.7.4.2.1.2.

### 15536 **10.7.4.2.2.4 ECC Implicit Certificates**



15537 The exact format of the 48-byte implicit certificate  $IC_U$  used with CBKE scheme shall be specified as follows:

15538  $IC_U = PublicReconstrKey \parallel Subject \parallel Issuer \parallel ProfileAttributeData$

15539 Where,

- 15540 1. *PublicReconstrKey*: the 22-byte representation of the public-key reconstruction data BEU as specified in the  
15541 implicit certificate generation protocol, which is an elliptic-curve point as specified in sub-clause 10.7.4.2.2.2  
15542 (see SEC4 [O1]);
- 15543 2. *Subject*: the 8-byte identifier of the entity  $U$  that is bound to the public-key reconstruction data  $BEU$  during  
15544 execution of the implicit certificate generation protocol (i.e., the extended, 64-bit IEEE 802.15.4 address  
15545 [E1] of the device that purportedly owns the private key corresponding to the public key that can be  
15546 reconstructed with *PublicReconstrKey*);
- 15547 3. *Issuer*: the 8-byte identifier of the CA that creates the implicit certificate during the execution of the implicit  
15548 certificate generation protocol (the so-called Certificate Authority).
- 15549 4. *ProfileAttributeData*: the 10-byte sequence of octets that can be used by a ZigBee profile for any purpose.  
15550 The first two bytes of this sequence is reserved as a profile identifier, which must be defined by another  
15551 ZigBee standard.
- 15552 5. The string  $I_U$  as specified in Step 6 of the actions of the CA in the implicit certificate generation protocol  
15553 (see section SEC4 [O3]) shall be the concatenation of the *Subject*, *Issuer*, and *ProfileAttributeData*:  
15554  $I_U = Subject \parallel Issuer \parallel ProfileAttributeData$

#### 15555 **10.7.4.2.2.5 Block-Cipher**

15556 The block-cipher used in this specification shall be the Advanced Encryption Standard AES-128, as specified in  
15557 FIPS Pub 197 [N4]. This block-cipher has a key size that is equal to the block size, in bits, i.e.,  $keylen = 128$ .

#### 15558 **10.7.4.2.2.6 Cryptographic Hash Function**

15559 The cryptographic hash function used in this specification shall be the blockcipher based cryptographic hash  
15560 function specified in Annex B.6 in [Z1], with the following instantiations:

- 15561 6. Each entity shall use the block-cipher  $E$  as specified in sub-clause B.1.1 in [Z1].
- 15562 7. All integers and octets shall be represented as specified in sub-clause 10.7.4.2.1.

15563 The Matyas-Meyer-Oseas hash function (specified in Annex B.6 in [Z1]) has a message digest size  $hashlen$  that is  
15564 equal to the block size, in bits, of the established blockcipher.

#### 15565 **10.7.4.2.2.7 Keyed Hash Function for Message Authentication**

15566 The keyed hash message authentication code (HMAC) used in this specification shall be HMAC, as specified in  
15567 the FIPS Pub 198 [N5] with the following instantiations:

- 15568 1. Each entity shall use the cryptographic hash  $H$  function as specified in sub-clause 10.7.4.2.2.6;
- 15569 2. The block size  $B$  shall have the integer value 16 (this block size specifies the length of the data integrity key,  
15570 in bytes, that is used by the keyed hash function, i.e., it uses a 128-bit data integrity key). This is also  
15571 *MacKeyLen*, the length of *MacKey*.
- 15572 3. The output size *HMAClen* of the HMAC function shall have the same integer value as the message digest  
15573 parameter *hashlen* as specified in sub-clause 10.7.4.2.2.6.

#### 15574 **10.7.4.2.2.8 Derived Shared Secret**

15575 The derived shared secret *KeyData* is the output of the key establishment. *KeyData* shall have length *KeyDataLen*  
15576 of 128 bits.

15577 **10.7.4.2.3 Certificate-Based Key-Establishment**

15578 The CBKE method is used when the authenticity of both parties involved has not been established and where  
15579 implicit authentication of both parties is required prior to key agreement.

15580 The CBKE protocol has an identical structure to the PKKE protocol, except that implicit certificates are used rather  
15581 than manual certificates. The implicit certificate protocol used with CBKE shall be the implicit certificate scheme  
15582 with associated implicit certificate generation scheme and implicit certificate processing transformation as specified  
15583 in SEC4 [O1], with the following instantiations:

- 15584       1. Each entity shall be a DEV;
- 15585       2. Each entity's identifier shall be its 64-bit device address [Z11]; the parameter *entlen* shall have the integer  
15586 value 64;
- 15587       3. Each entity shall use the cryptographic hash function as specified in sub-clause 10.7.4.2.2.6;

15588 The following additional information shall have been unambiguously established between devices operating the  
15589 implicit certificate scheme:

- 15590       1. Each entity shall have obtained information regarding the infrastructure that will be used for the operation of  
15591 the implicit certificate scheme - including a certificate format and certificate generation and processing rules  
15592 (see SEC4 [O1]);
- 15593       2. Each entity shall have access to an authentic copy of the elliptic-curve public keys of one or more certificate  
15594 authorities that act as CA for the implicit certificate scheme (SEC4 [O1]).

15595 The methods by which this information is to be established are outside the scope of this standard.

15596 The methods used during the CBKE protocol are described below. The parameters used by these methods are  
15597 described in Table 10-82.

15598

**Table 10-82. Parameters Used by Methods of the CBKE Protocol**

Parameter	Size (Octets)	Description
CERTU	48	The initiator device's implicit certificate used to transfer the initiator device's public key (denoted $Q_{I,U}$ in the Elliptic Curve MQV scheme in SEC1 [O1]) and the initiator device's identity.
CERTV	48	The responder device's implicit certificate used to transfer the responder device's public key (denoted $Q_{I,V}$ in the Elliptic Curve MQV scheme in SEC1 [O1]) and the responder device's identity.
QEU	22	The ephemeral public key generated by the initiator device (denoted $Q_{2,U}$ in the Elliptic Curve MQV scheme in SEC1 [O1]).
QEV	22	The ephemeral public key generated by the responder device (denoted $Q_{2,V}$ in the Elliptic Curve MQV scheme in SEC1 [O1]).
MACU	16	The secure message authentication code generated by the initiator device (where the message M is $(02)_{16}    ID_U    ID_V    QEU    QEV$ ) 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 10.7.4.2.2.2. See also section 3.7 of SEC1 [O1]).
MACV	16	The secure message authentication code generated by the responder device (where the message M is $(03)_{16}    ID_V    ID_U    QEV    QEU$ ) 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 10.7.4.2.2.3. See also section 3.7 of SEC1 [O1]).

15599 **10.7.4.2.3.1 Exchange Ephemeral Data**

15600 **10.7.4.2.3.1.1 Initiator**

15601 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.

15604 **10.7.4.2.3.1.2 Responder**

15605 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.

15608 **10.7.4.2.3.2 Validate Implicit Certificates**

15609 **10.7.4.2.3.2.1 Initiator**

15610 The initiator device's Key Establishment Cluster Client processes the Initiate Key Establishment response command.  
 15611 The initiator device examines CERTV (formatted as  $IC_V$  as described in sub-clause 10.7.4.2.2.4), confirms that the Subject identifier is the purported owner of the certificate, and runs the certificate processing steps described in section SEC4 [O2].  
 15612  
 15613

**15614 10.7.4.2.3.2 Responder**

15615 The responder device's Key Establishment Cluster Server processes the Initiate Key Establishment command. The  
15616 responder device examines CERTU (formatted as  $IC_U$  as described in sub-clause 10.7.4.2.2.4), confirms that the  
15617 Subject identifier is the purported owner of the certificate, and runs the certificate processing steps described in  
15618 section SEC 4 [O2].

**15619 10.7.4.2.3.3 Derive Keying Material****15620 10.7.4.2.3.3.1 Initiator**

15621 The initiator performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [O1] with the following  
15622 instantiations:

- 15623 1. The elliptic curve domain parameters shall be as specified in sub-clause 10.7.4.2.2.1;
- 15624 2. The KDF shall use the cryptographic hash function specified in sub-clause 10.7.4.2.2.2;
- 15625 3. The static public key  $Q_{1,U}$  shall be the static public key of the initiator;
- 15626 4. The ephemeral public key  $Q_{2,U}$  shall be an ephemeral public key of the initiator generated as part of this  
15627 transaction;
- 15628 5. The static public key  $Q_{1,V}$  shall be the static public key of the responder obtained from the responder's  
15629 certificate communicated to the initiator by the responder;
- 15630 6. The ephemeral public key  $Q_{2,V}$  shall be based on the point-compressed octet string representation  $QEV$  of an  
15631 ephemeral key of the responder communicated to the initiator by the responder;
- 15632 7. The KDF parameter *keydatalen* shall be *MacKeyLen* + *KeyDataLen*, where *MacKeyLen* is the length of  
15633 *MacKey* and *KeyDataLen* is the length of *KeyData*;
- 15634 8. The parameter *SharedInfo* shall be the empty string.

15635 The initiator device derives the keying material *MacKey* and *KeyData* from the output *K* as specified in section  
15636 3.6.1 of SEC1 [O1] by using *MacKey* as the leftmost *MacKeyLen* octets of *K* and *KeyData* as the rightmost  
15637 *KeyDataLen* octets of *K*. *KeyData* is used subsequently as the shared secret and *MacKey* is used for key  
15638 confirmation.

**15639 10.7.4.2.3.3.2 Responder**

15640 The responder performs the Elliptic Curve MQV scheme as specified in section 6.2 of SEC1 [O1] with the  
15641 following instantiations:

- 15642 1. The elliptic curve domain parameters shall be as specified in sub-clause 10.7.4.2.2.1;
- 15643 2. The KDF shall use the cryptographic hash function specified in sub-clause 10.7.4.2.2.2;
- 15644 3. The static public key  $Q_{1,U}$  shall be the static public key of the initiator obtained from the initiator's certificate  
15645 communicated to the responder by the initiator;
- 15646 4. The ephemeral public key  $Q_{2,U}$  shall be based on the point-compressed octet string representation  $QEU$  of  
15647 an ephemeral key of the initiator communicated to the responder by the initiator;
- 15648 5. The static public key  $Q_{1,V}$  shall be the static public key of the responder;
- 15649 6. The ephemeral public key  $Q_{2,V}$  shall be an ephemeral public key of the responder generated as part of this  
15650 transaction;
- 15651 7. The KDF parameter *keydatalen* shall be *MacKeyLen* + *KeyDataLen*, where *MacKeyLen* is the length of  
15652 *MacKey* and *KeyDataLen* is the length of *KeyData*;
- 15653 8. The parameter *SharedInfo* shall be the empty string.

15654 The responder device derives the keying material MacKey and KeyData from the output K as specified in section  
15655 3.6.1 of SEC1 [O1] by using MacKey as the leftmost MacKeyLen octets of K and KeyData as the rightmost  
15656 KeyDataLen octets of K. KeyData is used subsequently as the shared secret and MacKey is used for key  
15657 confirmation.

#### 15658 **10.7.4.2.3.4 Confirm Keys**

##### 15659 **10.7.4.2.3.4.1 Initiator**

15660 The initiator device uses MacKey to compute its message authentication code MACU and sends it to the  
15661 responder device by using the Confirm Key command via the Key Establishment Cluster Client.

15662 The initiator device uses MacKey to confirm the authenticity of the responder by calculating MACV and comparing  
15663 it with that sent by the responder.

##### 15664 **10.7.4.2.3.4.2 Responder**

15665 The responder device uses MacKey to compute its message authentication code MACV and sends it to the  
15666 initiator device by using the Confirm Key response command via the Key Establishment Cluster Server.

15667 The responder device uses MacKey to confirm the authenticity of the initiator by calculating MACU and comparing  
15668 it with that sent by the initiator.

### 15669 **10.7.5 Key Establishment Test Vectors**

15670 The following details the key establishment exchange data transformation and validation of test vectors for a  
15671 pair of Smart Energy devices using Certificate based key exchange (CBKE) using Elliptical Curve Cryptography  
15672 (ECC).

#### 15673 **10.7.5.1 Preconfigured Data**

15674 Each device is expected to have been preinstalled with security information prior to initiating key establishment.  
15675 The preinstalled data consists of the Certificate Authority's Public Key, a device specific certificate, and a device  
15676 specific private key.

##### 15677 **10.7.5.1.1 CA Public Key**

15678 The following is the Certificate Authority's Public Key.

15679 02 00 FD E8 A7 F3 D1 08

15680 42 24 96 2A 4E 7C 54 E6

15681 9A C3 F0 4D A6 B8

##### 15682 **10.7.5.1.2 Responder Data**

15683 The following is the certificate for device 1. The device has an IEEE of (>) 0000000000000001, and will be the  
15684 responder.

15685 03 04 5F DF C8 D8 5F FB

15686 8B 39 93 CB 72 DD CA A5

15687 5F 00 B3 E8 7D 6D 00 00

15688 00 00 00 00 00 01 54 45

15689 53 54 53 45 43 41 01 09

15690 00 06 00 00 00 00 00 00

15691

15692 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

15693

15694 The private key for device 1 is as follows:

15695 00 b8 a9 00 fc ad eb ab

15696 bf a3 83 b5 40 fc e9 ed

15697 43 83 95 ea a7

15698

15699 The public key for device 1 is as follows:

15700 03 02 90 a1 f5 c0 8d ad

15701 5f 29 45 e3 35 62 0c 7a

15702 98 fa c4 66 66 a1

### 15703 10.7.5.1.3 Initiator Data

15704 The following is the certificate for device 2. The device has an IEEE of (>) 0000000000000002, and will be the  
15705 initiator.

15706 02 06 15 E0 7D 30 EC A2

15707 DA D5 80 02 E6 67 D9 4B

15708 C1 B4 22 39 83 07 00 00

15709 00 00 00 00 00 02 54 45

15710 53 54 53 45 43 41 01 09

15711 00 06 00 00 00 00 00 00

15712

15713 The certificate has the following data embedded within it:

Public Key Reconstruction Data	02 06 15 E0 7D 30 EC A2 DA D5 80 02 E6 67 D9 -----
--------------------------------	---

Subject (IEEE)	00 00 00 00 00 00 00 02
Issuer	54 45 53 54 53 45 43 41
Attributes	01 09 00 06 00 00 00 00 00 00

15714

15715 The private key for device 2 is as follows:

15716 01 E9 DD B5 58 0C F7 2E

15717 CE 7F 21 5F 0A E5 94 E4

15718 8D F3 E7 FE E8

15719

15720 The public key for device 2 is:

15721 03 02 5B BA 38 D0 C7 B5

15722 43 6B 68 DF 72 8F 09 3E

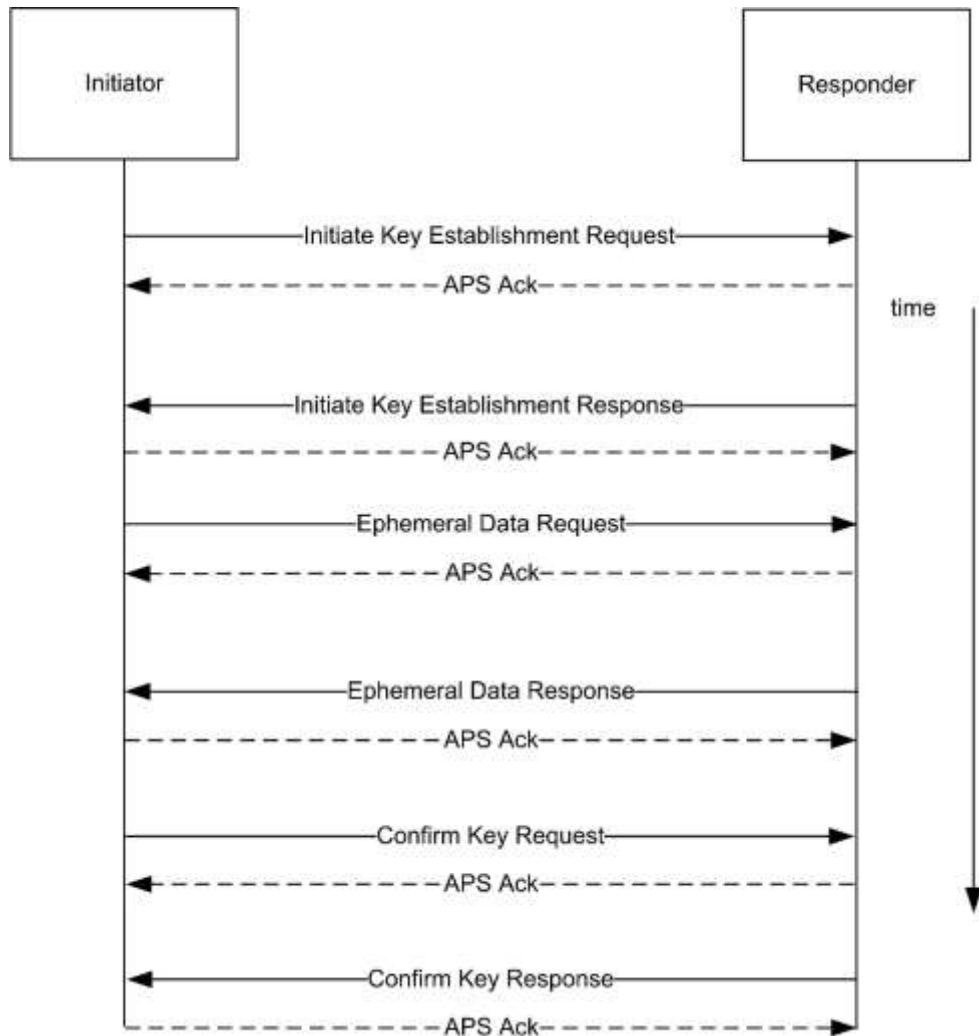
15723 7A 1D 6C 43 7E 6D

## 15724 **10.7.5.2 Key Establishment Messages**

15725 Figure 10-67 shows the basic flow of messages back and forth between the initiator and the responder performing  
15726 key establishment using the Key Establishment Cluster.

15727

**Figure 10-67. Key Establishment Command Exchange**



15728

15729 **10.7.5.2.1 Initiate Key Establishment Request**

15730 The following is the APS message sent by the initiator (device 2) to the responder (device 1) for the initiate key  
15731 establishment request.

15732 40 0A 00 08 09 01 0A 01

15733 01 00 00 01 00 03 06 02

15734 06 15 E0 7D 30 EC A2 DA

15735 D5 80 02 E6 67 D9 4B C1

15736 B4 22 39 83 07 00 00 00

15737 00 00 00 00 02 54 45 53

15738 54 53 45 43 41 01 09 00



15739 06 00 00 00 00 00 00

15740

15741 **APS Header**

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x01

15742

15743 **ZCL Header**

Frame Control	0x01	Client to Server
Sequence Number	0x00	
Command Identifier	0x00	<i>Initiate Key Establishment Request</i>
Key Establishment Suite	0x0001	ECMQV
Ephemeral Data Generate Time	0x03	
Confirm Key Generate Time	0x06	
Identity (IDU)	*	Device 2's certificate

15744 **10.7.5.2.2 Initiate Key Establishment Response**

15745 The following is the APS message sent by the responder (device 1) to the initiator (device 2) for the initiate key  
 15746 establishment response.

15747 40 0A 00 08 09 01 0A 01

15748 09 00 00 01 00 03 06 03

15749 04 5F DF C8 D8 5F FB 8B

15750 39 93 CB 72 DD CA A5 5F

15751 00 B3 E8 7D 6D 00 00 00

15752 00 00 00 00 01 54 45 53

15753 54 53 45 43 41 01 09 00

15754 06 00 00 00 00 00 00

15755

15756 **APS Header**

Frame Control	0x40
---------------	------

Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x01

15757

15758 **ZCL Header**

Frame Control	0x09	Server to Client
Sequence Number	0x00	
Command Identifier	0x00	<i>Initiate Key Establishment Response</i>
Key Establishment Suite	0x0001	ECMQV
Ephemeral Data Generate Time	0x03	
Confirm Key Generate Time	0x06	
Identity (IDV)	*	Device 1's certificate

15759 **10.7.5.2.3 Ephemeral Data Request**

15760 The following is the APS message sent by the initiator to the responder for the ephemeral data request.

15761 40 0A 00 08 09 01 0A 02

15762 01 01 01 03 00 E1 17 C8

15763 6D 0E 7C D1 28 B2 F3 4E

15764 90 76 CF F2 4A F4 6D 72

15765 88

15766

15767 **APS Header**

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

15768

15769 **ZCL Header**

Frame Control	0x01	Client to Server
Sequence Number	0x01	

Command Identifier	0x01	<i>Ephemeral Data Request</i>
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	

15770 **10.7.5.2.4 Ephemeral Data Response**

15771 The following is the APS message sent by the responder to the initiator for the ephemeral data response.

15772 40 0A 00 08 09 01 0A 02

15773 09 01 01 03 06 AB 52 06

15774 22 01 D9 95 B8 B8 59 1F

15775 3F 08 6A 3A 2E 21 4D 84

15776 5E

15777 **APS Header**

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

15778

15779 **ZCL Header**

Frame Control	0x09	Server to Client
Sequence Number	0x01	
Command Identifier	0x01	<i>Ephemeral Data Response</i>
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	

15780 **10.7.5.2.5 Confirm Key Request**

15781 The following is the APS message sent by the initiator to the responder for the confirm key request.

15782 40 0A 00 08 09 01 0A 03

15783 01 02 02 B8 2F 1F 97 74

15784 74 0C 32 F8 0F CF C3 92

15785 1B 64 20

15786

15787 **APS Header**

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

15788

15789 **ZCL Header**

Frame Control	0x01	Client to Server
Sequence Number	0x02	
Command Identifier	0x02	<i>Confirm Key Request</i>
Secure Message Authentication Code (MACU)	B8 2F 1F 97 74 74 0C 32 F8 0F CF C3 92 1B 64 20	

15790 **10.7.5.2.6 Confirm Key Response**

15791 The following is the APS message sent by the responder to the initiator for the confirm key response.

15792 40 0A 00 08 09 01 0A 03

15793 09 02 02 79 D5 F2 AD 1C

15794 31 D4 D1 EE 7C B7 19 AC

15795 68 3C 3C

15796

15797 **APS Header**

Frame Control	0x40
Destination Endpoint	0x0A
Cluster Identifier	0x0800
Profile ID	0x0109
Source Endpoint	0x0A
APS Counter	0x02

15798

15799 **ZCL Header**

Frame Control	0x09	Server to Client
Sequence Number	0x02	

Command Identifier	0x02	<i>Confirm Key Response</i>
Secure Message Authentication Code (MACV)	79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C	

15800 **10.7.5.3 Data Transformation**

15801 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

15802

15803 **Note:** '||' stands for bitwise concatenation

15804 **10.7.5.3.1 ECMQV Primitives**

15805 It is assumed that an ECC library is available for creating the shared secret given the local private key, local  
 15806 ephemeral public & private key, remote device's certificate, remote device's ephemeral public key, and the  
 15807 certificate authority's public key. Further it is assumed that this library has been separately validated with a set  
 15808 of ECC test vectors. Those test vectors are outside the scope of this document.

15809 **10.7.5.3.2 Key Derivation Function (KDF)**

15810 Once a shared secret (Z) is established, a transform is done to create a SMAC (Secure Message Authentication  
 15811 Code) and a shared ZigBee Key.

15812 **10.7.5.3.3 Initiator Transform**

15813 Upon receipt of the responder's ephemeral data response, the initiator has all the data necessary to calculate the  
15814 shared secret and derive the data for the confirm key request (SMAC).

15815 **10.7.5.3.3.1 Ephemeral Data**

Public Key	03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88
Private Key	00 13 D3 6D E4 B1 EA 8E 22 73 9C 38 13 70 82 3F 40 4B FF 88 62

15816 **10.7.5.3.3.2 Step Summary**

- 15817 1. Derive the Shared Secret using the ECMQV primitives  
15818  $Z = \text{ECC\_GenerateSharedSecret}(\text{Private}(U), E(U), E\text{-}P(U), \text{Cert}(V), E(V), CA)$   
15819 2. Derive the Keying data  
15820  $\text{Hash-1} = Z \parallel 00\ 00\ 00\ 01 \parallel \text{SharedData}$   
15821  $\text{Hash-2} = Z \parallel 00\ 00\ 00\ 02 \parallel \text{SharedData}$   
15822 3. Parse KeyingData as follows  
15823  $\text{MacKey} = \text{First 128 bits (Hash-1) of KeyingData}$   
15824  $\text{KeyData} = \text{Second 128 bits (Hash-2) of KeyingData}$   
15825 4. Create  $\text{MAC}(U)$   
15826  $\text{MAC}(U) = \text{MAC}(\text{MacKey}) \{ M(U) \parallel \text{ID}(U) \parallel \text{ID}(V) \parallel E(U) \parallel E(V) \}$   
15827 5. Send  $\text{MAC}(U)$  to V.  
15828 6. Receive  $\text{MAC}(V)$  from V.  
15829 7. Calculate  $\text{MAC}(V)'$   
15830  $\text{MAC}(V) = \text{MAC}(\text{MacKey}) \{ M(V) \parallel \text{ID}(V) \parallel \text{ID}(U) \parallel E(V) \parallel E(U) \}$   
15831 8. Verify  $\text{MAC}(V)'$  is the same as  $\text{MAC}(V)$ .

15832 **10.7.5.3.3.3 Detailed Steps**

- 15833 1. Derive the Shared Secret using the ECMQV primitives  
15834  $Z = \text{ECC\_GenerateSharedSecret}(\text{Private}(U), E(U), E\text{-}P(U), \text{Cert}(V), E(V), CA)$   
15835  $00\ E0\ D2\ C3\ CC\ D5\ C1\ 06\ A8\ 9C\ 4F\ 6C\ C2\ 6A\ 5F\ 7E$   
15836  $C9\ DF\ 78\ A7\ BE$   
15837 2. Derive the Keying data  
15838  $\text{Hash-1} = Z \parallel 00\ 00\ 00\ 01 \parallel \text{SharedData}$   
15839 **Concatenation**  
15840  $00\ E0\ D2\ C3\ CC\ D5\ C1\ 06\ A8\ 9C\ 4F\ 6C\ C2\ 6A\ 5F\ 7E$   
15841  $C9\ DF\ 78\ A7\ BE\ 00\ 00\ 00\ 01$

15842       **Hash**  
 15843               90 F9 67 B2 2C 83 57 C1 0C 1C 04 78 8D E9 E8 48

15844               Hash-2 = Z || 00 00 00 02 || SharedData

15845       **Concatenation**  
 15846               00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E  
 15847               C9 DF 78 A7 BE 00 00 00 02

15848       **Hash**  
 15849               86 D5 8A AA 99 8E 2F AE FA F9 FE F4 96 06 54 3A

15850       3. Parse KeyingData as follows  
 15851               MacKey = First 128 bits (Hash-1) of KeyingData  
 15852               KeyData = Second 128 bits (Hash-2) of KeyingData

15853       4. Create MAC(U)  
 15854               MAC(U) = MAC(MacKey) { M(U) || ID(U) || ID(V) || E(U) || E(V) }

15855       **Concatenation**  
 15856               02 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00  
 15857               01 03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76  
 15858               CF F2 4A F4 6D 72 88 03 06 AB 52 06 22 01 D9 95  
 15859               B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E 88 00 10

15860       **Hash**  
 15861               B8 2F 1F 97 74 74 0C 32 F8 0F CF C3 92 1B 64 20

15862       5. Send MAC(U) to V.  
 15863       6. Receive MAC(V) from V.  
 15864       7. Calculate MAC(V)'  
 15865               MAC(V) = MAC(MacKey) { M(V) || ID(V) || ID(U) || E(V) || E(U) }

15866       **Concatenation**  
 15867               03 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  
 15868               02 03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08  
 15869               6A 3A 2E 21 4D 84 5E 03 00 E1 17 C8 6D 0E 7C D1  
 15870               28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88 88 00 10

15871       **Hash**  
 15872               79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C

15873       8. Verify MAC(V)' is the same as MAC(V).

15874 **10.7.5.3.4 Responder Transform**

15875 Upon receipt of the initiator's confirm key request, the responder has all the data necessary to calculate the shared  
 15876 secret, validate the initiator's confirm key message, and derive the data for the confirm key response (SMAC).

15877 **10.7.5.3.4.1 Ephemeral Data**

Public Key	03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E
Private Key	03 D4 8C 72 10 DD BC C4 FB 2E 5E 7A 0A A1 6A 0D B8 95 40 82 0B

15878 **10.7.5.3.4.2 Step Summary**

- 15879 1. Derive the Shared Secret using the ECMQV primitives  
 15880  $Z = \text{ECC\_GenerateSharedSecret}(\text{Private}(V), E(V), E\text{-}P(V), \text{Cert}(U), E(U), CA)$   
 15881 2. Derive the Keying data  
 15882  $\text{Hash-1} = Z \parallel 00\ 00\ 00\ 01 \parallel \text{SharedData}$   
 15883  $\text{Hash-2} = Z \parallel 00\ 00\ 00\ 02 \parallel \text{SharedData}$   
 15884 3. Parse KeyingData as follows  
 15885  $\text{MacKey} = \text{First 128 bits (Hash-1) of KeyingData}$   
 15886  $\text{KeyData} = \text{Second 128 bits (Hash-2) of KeyingData}$   
 15887 4. Create  $\text{MAC}(V)$   
 15888  $\text{MAC}(V) = \text{MAC}(\text{MacKey}) \{ M(V) \parallel \text{ID}(V) \parallel \text{ID}(U) \parallel E(V) \parallel E(U) \}$   
 15889 5. Calculate  $\text{MAC}(U)$   
 15890  $\text{MAC}(U) = \text{MAC}(\text{MacKey}) \{ M(U) \parallel \text{ID}(U) \parallel \text{ID}(V) \parallel E(U) \parallel E(V) \}$   
 15891 6. Verify  $\text{MAC}(U)$  is the same as  $\text{MAC}(U)$ .  
 15892 7. Send  $\text{MAC}(V)$  to U.

15893 **10.7.5.3.4.3 Detailed Steps**

- 15894 8. Derive the Shared Secret using the ECMQV primitives  
 15895  $Z = \text{ECC\_GenerateSharedSecret}(\text{Private}(U), E(U), E\text{-}P(U), \text{Cert}(V), E(V), CA)$   
 15896 00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E  
 15897 C9 DF 78 A7 BE

- 15898 9. Derive the Keying data  
 15899  $\text{Hash-1} = Z \parallel 00\ 00\ 00\ 01 \parallel \text{SharedData}$

15900 **Concatenation**

15901 00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E  
 15902 C9 DF 78 A7 BE 00 00 00 01

15903 **Hash**

15904 90 F9 67 B2 2C 83 57 C1 0C 1C 04 78 8D E9 E8 48



15905 Hash-2 = Z || 00 00 00 02 || SharedData

15906 **Concatenation**

15907 00 E0 D2 C3 CC D5 C1 06 A8 9C 4F 6C C2 6A 5F 7E

15908 C9 DF 78 A7 BE 00 00 00 02

15909 **Hash**

15910 86 D5 8A AA 99 8E 2F AE FA F9 FE F4 96 06 54 3A

15911 10. Parse KeyingData as follows

15912 MacKey = First 128 bits (Hash-1) of KeyingData

15913 KeyData = Second 128 bits (Hash-2) of KeyingData

15914 11. Create MAC(V)

15915  $MAC(V) = MAC(MacKey) \{ M(V) || ID(V) || ID(U) || E(V) || E(U) \}$

15916 **Concatenation**

15917 03 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00

15918 02 03 06 AB 52 06 22 01 D9 95 B8 B8 59 1F 3F 08

15919 6A 3A 2E 21 4D 84 5E 03 00 E1 17 C8 6D 0E 7C D1

15920 28 B2 F3 4E 90 76 CF F2 4A F4 6D 72 88 88 00 10

15921 **Hash**

15922 79 D5 F2 AD 1C 31 D4 D1 EE 7C B7 19 AC 68 3C 3C

15923 12. Calculate MAC(V)'

15924  $MAC(U) = MAC(MacKey) \{ M(U) || ID(U) || ID(V) || E(U) || E(V) \}$

15925 **Concatenation**

15926 02 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00

15927 01 03 00 E1 17 C8 6D 0E 7C D1 28 B2 F3 4E 90 76

15928 CF F2 4A F4 6D 72 88 03 06 AB 52 06 22 01 D9 95

15929 B8 B8 59 1F 3F 08 6A 3A 2E 21 4D 84 5E 88 00 10

15930 **Hash**

15931 B8 2F 1F 97 74 74 0C 32 F8 0F CF C3 92 1B 64 20

15932 13. Verify MAC(V) is the same as MAC(V).

15933 14. Send MAC(V) to U.

15934



## CHAPTER 11 OVER-THE-AIR UPGRADE

15935

15936 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
15937 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
15938 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
15939 using [*Rn*] notation.

### 11.1 Introduction

15940

#### 11.1.1 Purpose

15941

15942 The objective of this chapter is to provide detailed technical requirements for Over-The-Air image upgrade. This  
15943 chapter presents a clear methodology for implementation of the OTA Upgrade cluster using the existing ZigBee  
15944 stack(s), ZigBee Cluster Library and this OTA cluster.

15945 The main goal of Over-The-Air Upgrade cluster is to provide an interoperable means for devices from different  
15946 manufacturers to upgrade each other's image. Additionally, the OTA Upgrade cluster defines a mechanism by which  
15947 security credentials, logs and configuration file types are accessible by offering a solution that utilizes a set of optional  
15948 and mandatory commands.

#### 11.1.2 Scope

15949

15950 This chapter will only describe features that require implementation in order to be ZigBee OTA upgrade (cluster)  
15951 certified. Other optional features including using multicast for sending upgrade messages and (upgrade) cloning will  
15952 not be discussed in this document.

15953 Currently, only Application Bootloader support is required in order to support ZigBee OTA Upgrade cluster. MAC  
15954 Bootloader upgrading is not supported at the moment.

#### 11.1.3 Terminology

15955

15956 **Application Standard or Standard** – This is a noun that refers to any application standard specification that includes  
15957 this specification. Examples: ZigBee Home Automation, ZigBee Smart Energy, etc.

### 11.2 General Description

15958

#### 11.2.1 Introduction

15959

15960 The existing OTA upgrade methods available are platform specific, not OTA interoperable and do not provide a  
15961 common framework for upgrading networks that support a mix of devices from multiple platforms and ZigBee Stack  
15962 vendors.

15963 The intent of this chapter is to provide an interoperable OTA upgrade of new image for devices deployed in the field.  
15964 As long as the device supports the OTA Upgrade cluster and it is certified by an approved test house, its image SHALL  
15965 be upgradeable by another device from the same or different manufacturer that also implemented and certified the  
15966 OTA Upgrade cluster.

15967 OTA Upgrade cluster will also require that in order to support OTA upgrade, the device will need to have an  
15968 application bootloader installed as well as sufficient memory (external or internal) to store the newly loaded image.  
15969 An application bootloader uses the running ZigBee stack and application to retrieve and store a new image. Depending  
15970 upon the manufacturer, the image MAY consist of a bootloader image, a ZigBee stack image or only a patch to the  
15971 application image. Whatever comprises the OTA upgrade image being sent to the node does not concern the ZigBee  
15972 OTA cluster and it is outside the scope.

15973 To use an application bootloader, the device is required to have sufficient memory (internal or external) to store the  
 15974 newly downloaded OTA upgrade image. By doing so, the current running image is not overwritten until the new image  
 15975 has been successfully downloaded. It also allows the possibility of a node saving an image in its memory and  
 15976 forwarding that image to another node. Application bootloading provides flexibility of when the device decides to  
 15977 download new OTA upgrade image as well as when the device decides to switch to running the new image.

15978 Since the bootloading is done at the application level, it automatically makes use of various features already offered  
 15979 by the ZigBee Network Layer and Application Sub Layer (APS) including the ability to bootload a device that is  
 15980 multiple hops away, message retries to increase reliability, and security. It also allows the network to continue to  
 15981 operate normally while the bootload is in progress. In addition, it supports bootloading of sleeping  
 15982 (RxOnWhenIdle=FALSE) devices.

15983 The application bootload messages are built upon typical ZigBee messages, with additional ZigBee Cluster Library  
 15984 (ZCL) header and payload and ZigBee OTA cluster specific payload.

15985 An application standard that includes this specification MAY, of course, add requirements and  
 15986 dependencies not defined here. Such a standard SHALL not relax requirements by changing a feature  
 15987 here from mandatory to optional, but it MAY specify features it deems mandatory, that are optional in this  
 15988 specification.

15989

15990 For example: The ZigBee Smart Energy standard has particular OTA cluster security feature  
 15991 requirements that are defined as mandatory in the ZSE specification, but are optional here.

## 11.2.2 Cluster List

15992

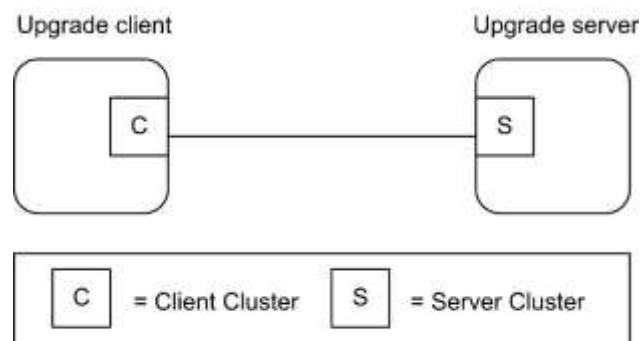
15993 The clusters defined in this document are listed in Table 11-1.

15994 **Table 11-1. Clusters Specified in This Document**

Cluster ID	Cluster Name	Description
0x0019	OTA Upgrade	Parameters and commands for upgrading image on devices Over-The-Air.

15995

15996 **Figure 11-1. Typical Usage of OTA Upgrade Cluster**



*Note: Device names are examples for illustration purposes only*

15997

15998

15999 Upgrade Client is a device to be upgraded with new image. Upgrade server is a device that has the new image to send  
 16000 to the client. This document SHALL specify how the client discovers the server, the over the air message format  
 16001 between the client and server, and the means for the server to signal the client to switch to running the new image.

16002 It is possible that the upgrade server MAY have several OTA upgrade images from different manufacturers. How the  
 16003 upgrade server receives these OTA upgrade images and how it stores and manages them are outside the scope of this  
 16004 document.

16005 In addition to the typical use case of transferring new firmware images to client devices, OTA Upgrade cluster MAY  
 16006 also be used to transfer device specific file types such as log, configuration or security credentials (needed for  
 16007 upgrading from SE 1.0 or SE 1.1 to SE 2.0). The cluster provides flexibility in OTA header and a set of optional  
 16008 commands that make transferring of such file types possible.

## 16009 11.3 OTA Upgrade

### 16010 11.3.1 Overview

16011 Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification,  
 16012 identification, etc.

16013 The cluster provides a standard way to upgrade devices in the network via OTA messages. Thus the upgrade process  
 16014 MAY be performed between two devices from different manufacturers. Devices are required to have application  
 16015 bootloader and additional memory space in order to successfully implement the cluster.

16016 It is the responsibility of the server to indicate to the clients when update images are available. The client MAY be  
 16017 upgraded or downgraded. The upgrade server knows which client devices to upgrade and to what file version. The  
 16018 upgrade server MAY be notified of such information via the backend system. For ZR clients, the server MAY send a  
 16019 message to notify the device when an updated image is available. It is assumed that ZED clients will not be awake to  
 16020 receive an unsolicited notification of an available image. All clients (ZR and ZED) SHALL query (poll) the server  
 16021 periodically to determine whether the server has an image update for them. Image Notify is optional.

16022 The cluster is implemented in such a way that the client service works on both ZED and ZR devices. Being able to  
 16023 handle polling is mandatory for all server devices while being able to send a notify is optional. Hence, all client devices  
 16024 must be able to use a ‘poll’ mechanism to send query message to the server in order to see if the server has any new  
 16025 file for it. The polling mechanism also puts fewer resources on the upgrade server. It is ideal to have the server maintain  
 16026 as little state as possible since this will scale when there are hundreds of clients in the network. The upgrade server is  
 16027 not required to keep track of what pieces of an image that a particular client has received; instead the client SHALL  
 16028 do that. Lastly poll makes more sense for devices that MAY need to perform special setup to get ready to receive an  
 16029 image, such as unlocking flash or allocating space for the new image.

#### 16030 11.3.1.1 Revision History

16031 The global mandatory *ClusterRevision* attribute SHALL reflect the revision of the implemented cluster specification  
 16032 as identified by one or more cluster identifiers listed below in this specification (see 2.3.5.1).

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCBs 1374 1470 1477 1540 1594 2046 2056
2	alternative Image Activation Policies; 128-bit Crypto suite, Smart Energy Profile 1.2a & 1.2b
3	CCB 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2296 2307 2315 2342 2398 2464

#### 16033 11.3.1.2 Classification

Hierarchy	Role	PICS Code

Base	Utility <sup>124</sup>	OTA
------	------------------------	-----

### 16034 11.3.1.3 Cluster Identifiers

Identifier	PICS Code	Name
0x0019	OTA	OTA Upgrade

## 16035 11.3.2 Security

16036 Security for the OTA Upgrade cluster encompasses these areas: image verification, image transport, and image  
 16037 encryption. Security mechanisms in the application standard dictate the security level of over-the-air image upgrading.  
 16038 For example, an application standard with strict security policies (such as Smart Energy) MAY support image  
 16039 signature as well as encryption in both network and APS layers; while other application standards MAY only support  
 16040 network encryption. Each application standard must decide the list of required security policies for their use of the  
 16041 OTA Upgrade cluster.

### 16042 11.3.2.1 Terminology

16043 There are many aspects to security. These can be broken down into the following areas: confidentiality, integrity,  
 16044 authentication, availability, and non-repudiation. Authorization and auditing can also be considered as part of security.

16045 **Confidentiality** – This is the requirement that no third party can read data that is not intended for that party. This is  
 16046 discussed below in Image Encryption.

16047 **Integrity** – This is the property of security where the recipient can verify that data was not modified between the time  
 16048 it was initially distributed by the sender and when it was received by the intended recipient. Modifications to the data  
 16049 by third parties can be detected. This is discussed in Image Verification below.

16050 **Authentication** – This is the property where the identity of the sender of data can be verified by the intended recipient.  
 16051 This is discussed in Image Verification below.

16052 **Availability** – This refers to the property that resources are available when they are required and cannot be unfairly  
 16053 consumed by an attacker. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

16054 **Non-repudiation** – This refers to the property where a sender/receiver cannot deny that a security exchange took  
 16055 place. The OTA Upgrade cluster does not address this; as such it will not be discussed any further.

## 16056 11.3.3 Image Verification

### 16057 11.3.3.1 Asymmetric Verification of Authenticity and Integrity

16058 It is strongly encouraged that there is a means to verify the authenticity and integrity of the bootload image. This is  
 16059 most often accomplished through asymmetric encryption technologies (i.e. public/private keys) where only one device  
 16060 is able to create a digital signature but many devices are able to verify it. Bootload images MAY be signed by the  
 16061 private key of the manufacturer with that signature appended to the image that is transported to the device. Once the  
 16062 complete image has been received the signature is verified using the public key of the signer.

16063 Devices MAY be pre-installed with the certificate (public key) of the device that created the signature, or they MAY  
 16064 receive the certificate over-the-air. How the signer's security data is obtained is considered outside the scope of the  
 16065 OTA Upgrade cluster and is manufacturer specific. When signer certificates are sent over-the-air and not pre-installed,  
 16066 it is recommended that the transportation of the certificate be done using encryption from a trusted source to reduce  
 16067 the chance an attacker MAY inject their own signer certificate into the device.

<sup>124</sup> CCB 2315 OTA is a Utility, not an Application class cluster

16068 Images with verification mechanisms built in MAY be transported over insecure communication mechanisms while  
16069 still maintaining their authenticity and integrity. In fact, it is likely that the originator of the upgrade image (the  
16070 manufacturer) will not be directly connected to any ZigBee networks and therefore distribute the upgrade image across  
16071 other mediums (such as the internet) before arriving on the ZigBee network. In that case it is crucial that the Image  
16072 Verification be independent of the communication medium. Any attempts to tamper with the signature or the data  
16073 itself must be detected and will cause the upgrade image to be rejected by the target device. An attacker that crafts its  
16074 own signed image and tries to have it accepted will be rejected since that image will not be signed by the  
16075 manufacturer's signing authority.

16076 It is up to each application standard to determine the minimum requirements for image verification. For example: for  
16077 the ZigBee Smart Energy standard there is already a minimum set of security requirements included in NEMA SG-  
16078 AMI 1-2009. The OTA Upgrade cluster will communicate the methods in use for basic image verification. Individual  
16079 manufacturers are free to augment this and provide their own extensions. Those extensions are outside the scope of  
16080 the OTA Upgrade cluster.

16081 Without asymmetric encryption technology, a device is limited in its ability to authenticate images. Images MAY be  
16082 encrypted with symmetric keys such that only those devices that need to decrypt the image have access to the key.  
16083 However, the security of this system is dependent on the security of all devices that have access to the symmetric key.

### 16084 **11.3.3.2 Verification of Integrity by Hash Value**

16085 For application standards that do not require the asymmetric verification method, the authenticity of the OTA image  
16086 cannot be verified. However, it is possible to verify the integrity of the OTA image by using the hash value method in  
16087 section 11.7.2. There will be no signer certificate nor any signature involved.

## 16088 **11.3.4 Image Transport**

16089 When there is a means to verify the authenticity of the bootload images, transport of the images in a secure fashion  
16090 provides little additional security to the integrity and authenticity. What secure transportation provides is a means to  
16091 communicate the policies about *when* a device SHOULD perform upgrade or *what version* it SHOULD upgrade to.

16092 A secured ZigBee network uses network security for all messages, but that does not provide point-to-point security.  
16093 APS security SHOULD be used to assure that messages are sent from only the trusted source (the upgrade server).  
16094 This will be utilized to provide implicit and explicit authorization by the upgrade server about when devices will  
16095 *initiate* bootload events.

16096 Distribution of upgrade image via broadcast or multicast messages is not recommended because its lack of point-to-  
16097 point security. Reception of upgrade images via broadcast or multicast SHOULD not be inferred as authorization by  
16098 the upgrade server to initiate the upgrade. In this case, the act of receiving the image and upgrading it SHOULD be  
16099 split up into separate events. The latter communications SHOULD be done via unicast to verify that the upgrade server  
16100 has authorized a device to upgrade to a previously received image. Applications must determine what level of  
16101 authorization is required by the upgrade server.

### 16102 **11.3.5 Image Signature**

16103 An application standard MAY require that the OTA Upgrade cluster provides mechanisms to sign the OTA file to  
16104 protect the authenticity and integrity of the image.

### 16105 **11.3.6 Image Integrity Code**

16106 An application standard MAY require that the OTA Upgrade cluster provides hash mechanisms to provide protection  
16107 against unintended data corruption.

## 16108 11.4 OTA File Format

### 16109 11.4.1 General Structure

16110 The OTA file format is composed of a header followed by a number of sub-elements. The header describes general  
16111 information about the file such as version, the manufacturer that created it, and the device it is intended for. Sub-  
16112 elements in the file MAY contain upgrade data for the embedded device, certificates, configuration data, log messages,  
16113 or other manufacturer specific pieces. Below is an example file.

16114 **Figure 11-2. Sample OTA File**

Octets	Variable	Variable	Variable	Variable
Data	OTA Header	Upgrade Image	Signer Certificate	Signature

16115  
16116 The OTA header will not describe details of the particular sub-elements. Each sub-element is self-describing. With  
16117 exception of a few sub-elements, the interpretation of the data contained is up to the manufacturer of the device.

### 16118 11.4.2 OTA Header Format

16119 **Table 11-2. OTA Header Fields**

Octets	Data Types	Field Names	M/O
4	uint32	OTA upgrade file identifier	M
2	uint16	OTA Header version	M
2	uint16	OTA Header length	M
2	map16 <sup>125</sup>	OTA Header Field control	M
2	uint16	Manufacturer code	M
2	uint16	Image type	M
4	uint32	File version	M
2	uint16	ZigBee Stack version	M
32	ASCII <sup>126</sup>	OTA Header string	M
4	uint32	Total Image size (including header)	M
0/1	uint8	Security credential version	O
0/8	EUI64	Upgrade file destination	O
0/2	uint16	Minimum hardware version	O
0/2	uint16	Maximum hardware version	O

<sup>125</sup> CCB 2219 Field Control is 16-bit bitmap, not an unsigned 16-bit integer

<sup>126</sup> this does not have a length byte, so it is not a character string data type



16120 The first entry of the table above (OTA upgrade file identifier) represents the first field in the OTA header, and the  
 16121 last entry represents the last field. The endianness used in each data field SHALL be little endian in order to be  
 16122 compliant with general ZigBee messages.

16123 Please refer to Chapter 2, Foundation Specification, for more description on data types.

### 11.4.2.1 OTA Upgrade File Identifier

16124 The value is a unique 4-byte value that is included at the beginning of all ZigBee OTA upgrade image files in order  
 16125 to quickly identify and distinguish the file as being a ZigBee OTA cluster upgrade file, without having to examine the  
 16126 whole file content. This helps distinguishing the file from other file types on disk. The value is defined to be  
 16127 “0x0BEEF11E”.

### 11.4.2.2 OTA Header Version

16129 The value enumerates the version of the header and provides compatibility information. The value is composed of a  
 16130 major and minor version number (one byte each). The high byte (or the most significant byte) represents the major  
 16131 version and the low byte (or the least significant byte) represents the minor version number. A change to the minor  
 16132 version means the OTA upgrade file format is still backward compatible, while a change to the major version suggests  
 16133 incompatibility.

16134 The current OTA header version SHALL be 0x0100 with major version of “01” and minor version of “00”.

### 11.4.2.3 OTA Header Length

16136 This value indicates full length of the OTA header in bytes, including the OTA upgrade file identifier, OTA header  
 16137 length itself to any optional fields. The value insulates existing software against new fields that MAY be added to the  
 16138 header. If new header fields added are not compatible with current running software, the implementations SHOULD  
 16139 process all fields they understand and then skip over any remaining bytes in the header to process the image or signing  
 16140 certificate. The value of the header length depends on the value of the OTA header field control, which dictates which  
 16141 optional OTA header fields are included.

### 11.4.2.4 OTA Header Field Control

16143 The bit mask indicates whether additional information such as Image Signature or Signing Certificate are included as  
 16144 part of the OTA Upgrade Image.

Table 11-3. OTA Header Field Control Bitmask

Bits	Name
0	Security Credential Version Present
1	Device Specific File
2	Hardware Versions Present

16147 Security credential version present bit indicates whether security credential version field is present or not in the OTA  
 16148 header.

16149 Device specific file bit in the field control indicates that this particular OTA upgrade file is specific to a single device.

16150 Hardware version present bit indicates whether minimum and maximum hardware version fields are present in the  
 16151 OTA header or not.

### 16152 11.4.2.5 Manufacturer Code

16153 This is the ZigBee assigned identifier for each member company. When used during the OTA upgrade process,  
16154 manufacturer code value of 0xffff has a special meaning of a wild card. The value has a ‘match all’ effect. OTA server  
16155 MAY send a command with wild card value for manufacturer code to match all client devices from all manufacturers.

### 16156 11.4.2.6 Image Type

16157 The manufacturer SHOULD assign an appropriate and unique image type value to each of its devices in order to  
16158 distinguish the products. This is a manufacturer specific value. However, the OTA Upgrade cluster has reserved the  
16159 last 64 values of image type value to indicate specific file types such as security credential, log, and configuration.  
16160 When a client wants to request one of these specific file types, it SHALL use one of the reserved image type values  
16161 instead of its own (manufacturer specific) value when requesting the image via Query Next Image Request command.

16162 **Table 11-4. Image Type Values**

File Type Values	File Type Description
0x0000 – 0xffbf	Manufacturer Specific
0xffc0	Client Security credentials
0xffc1	Client Configuration
0xffc2	Server Log
0xffc3	Picture
0xffff	wild card

16163  
16164 Image type value of 0xffff has a special meaning of a wild card. The value has a ‘match all’ effect. For example, the  
16165 OTA server MAY send Image Notify command with image type value of 0xffff to indicate to a group of client devices  
16166 that it has all types of images for the clients. Additionally, the OTA server MAY send Upgrade End Response  
16167 command with image type value of 0xffff to indicate a group of clients, with disregard to their image types, to upgrade.

### 16168 11.4.2.7 File Version

16169 For firmware image, the file version represents the release and build number of the image’s application and stack. The  
16170 application release and build numbers are manufacturer specific, however, each manufacturer SHOULD obtain stack  
16171 release and build numbers from their stack vendor. OTA Upgrade cluster makes the recommendation below regarding  
16172 how the file version SHOULD be defined, in an attempt to make it easy for humans and upgrade servers to determine  
16173 which versions are newer than others. The upgrade server SHOULD use this version value to compare with the one  
16174 received from the client.

16175 The server MAY implement more sophisticated policies to determine whether to upgrade the client based on the file  
16176 version. A higher file version number indicates a newer file.

16177 **Table 11-5. Recommended File Version Definition**

Application Release	Application Build	Stack Release	Stack Build
1 byte	1 byte	1 byte	1 byte
8-bit integer	8-bit integer	8-bit integer	8-bit integer

16178 For example,

- 16179 File version A: 0x10053519 represents application release 1.0 build 05 with stack release 3.5 b19.
- 16180 File version B: 0x10103519 represents application release 1.0 build 10 with stack release 3.5 b19.
- 16181 File version C: 0x10103701 represents application release 1.0 build 10 with stack release 3.7 b01.
- 16182 File version B is newer than File version A because its application version is higher, while File version C is newer than File version B because its stack version is higher.
- 16183
- 16184 The file version value MAY be defined differently for different image types. For example, version scheme for security credential data MAY be different than that of log or configuration file or a normal firmware upgrade image version.
- 16185
- 16186 The specific implementation of a versioning scheme is manufacturer specific.
- 16187 Note that a binary-coded decimal convention (BCD) concept is used here for version number. This is to allow easy conversion to decimal digits for printing or display, and allows faster decimal calculations.
- 16188

16189 **11.4.2.8 ZigBee Stack Version**

16190 This information indicates the ZigBee stack version that is used by the application. This provides the upgrade server an ability to coordinate the distribution of images to devices when the upgrades will cause a major jump that usually breaks the over-the-air compatibility, for example, from ZigBee Pro to upcoming ZigBee IP. The values below represent currently available ZigBee stack versions.

16194 **Table 11-6. ZigBee Stack Version Values**

ZigBee Stack Version Values	Stack Name
0x0000	ZigBee 2006
0x0001	ZigBee 2007
0x0002	ZigBee Pro
0x0003	ZigBee IP

16195 **11.4.2.9 OTA Header String**

16196 This is a manufacturer specific string that MAY be used to store other necessary information as seen fit by each manufacturer. The string SHALL be a null terminated string using ASCII encoding. Any bytes after the terminating character MAY be used by manufacturers for additional data transport and SHALL not be interpreted as human readable data. The idea is to have a human readable string that can prove helpful during development cycle. The string is defined to occupy 32 bytes of space in the OTA header.

16201 **11.4.2.10 Total Image Size**

16202 The value represents the total image size in bytes. This is the total of data in bytes that SHALL be transferred over-the-air from the server to the client. In most cases, the total image size of an OTA upgrade image file is the sum of the OTA header and the actual file data (along with its tag) lengths. If the image is a signed image and contains a certificate of the signer, then the Total image size SHALL also include the signer certificate and the signature (along with their tags) in bytes.

16207 This value is crucial in the OTA upgrade process. It allows the client to determine how many image request commands to send to the server to complete the upgrade process.

16209 **11.4.2.11 Security Credential Version**

16210 This information indicates security credential version type, such as SE1.0 or SE2.0 that the client is required to have,  
16211 before it SHALL install the image. One use case for this is so that after the client has downloaded a new image from  
16212 the server, it SHOULD check if the value of security credential version allows for running the image. If the client’s  
16213 existing security credential version does not match or is outdated from what specified in the OTA header, it SHOULD  
16214 obtain new security credentials before upgrading to running the new image.

16215 **Table 11-7. Security Credential Version**

Security Credential Version Values	Security Credential Version Types
0x00	SE 1.0
0x01	SE 1.1
0x02	SE 2.0
0x03	SE 1.2

16216 **11.4.2.12 Upgrade File Destination**

16217 If Device Specific File bit is set, it indicates that this OTA file contains security credential/certificate data or other  
16218 type of information that is specific to a particular device. Hence, the upgrade file destination field (in OTA header)  
16219 SHOULD also be set to indicate the IEEE address of the client device that this file is meant for.

16220 **11.4.2.13 Minimum Hardware Version**

16221 The value represents the earliest hardware platform version this image SHOULD be used on. This field is defined as  
16222 follows:

16223 **Table 11-8. Hardware Version Format**

Version	Revision
1 byte	1 byte
8-bit integer	8-bit integer

16224  
16225 The high byte represents the version and the low byte represents the revision.

16226 **11.4.2.14 Maximum Hardware Version**

16227 The value represents the latest hardware platform this image SHOULD be used on. The field is defined the same as  
16228 the Minimum Hardware Version (above).

16229 The hardware version of the device SHOULD not be earlier than the minimum (hardware) version and SHOULD not  
16230 be later than the maximum (hardware) version in order to run the OTA upgrade file.

16231 **11.4.3 Sub-element Format**

16232 Sub-elements in the file are composed of an identifier followed by a length field, followed by the data. The identifier  
 16233 provides for forward and backward compatibility as new sub-elements are introduced. Existing devices that do not  
 16234 understand newer sub-elements MAY ignore the data.

16235 **Figure 11-3. Sub-element Format**

Octets	2-bytes	4-bytes	Variable
Data	Tag ID	Length Field	Data

16236  
 16237 Sub-elements provide a mechanism to denote separate sections of data utilized by the device for the upgrade. For  
 16238 example, a device that has multiple processors each with their own firmware image could use a separate sub-element  
 16239 for each one. The details of how this is handled would be up to the manufacturer of the device.

16240 A few sub-elements are not manufacturer-specific and defined by the OTA cluster itself. See section 11.4.4 below.

16241 **11.4.3.1 Tag ID**

16242 The tag identifier denotes the type and format of the data contained within the sub-element. The identifier is one of  
 16243 the values from Table 11-9 below.

16244 **11.4.3.2 Length Field**

16245 This value dictates the length of the rest of the data within the sub-element in bytes. It does not include the size of the  
 16246 Tag ID or the Length Fields.

16247 **11.4.3.3 Data**

16248 The length of the data in the sub-element must be equal to the value of the Length Field in bytes. The type and format  
 16249 of the data contained in the sub-element is specific to the Tag.

16250 **11.4.4 Tag Identifiers**

16251 Sub-elements are generally specific to the manufacturer and the implementation. However, this specification has  
 16252 defined a number of common identifiers that MAY be used across multiple manufacturers.

16253 **Table 11-9. Tag Identifiers**

Tag Identifiers	Description
0x0000	Upgrade Image
0x0001	ECDSA Signature (Crypto Suite 1)
0x0002	ECDSA Signing Certificate (Crypto Suite 1)
0x0003	Image Integrity Code
0x0004	Picture Data
0x0005	ECDSA Signature (Crypto Suite 2)

Tag Identifiers	Description
0x0006	ECDSA Signing Certificate (Crypto Suite 2)
0xf000 – 0xffff	Manufacturer Specific Use

16254  
16255 Manufacturers MAY define tag identifiers for their own use and dictate the format and behavior of devices that receive  
16256 images with that data.

## 16257 11.4.5 Crypto Suites

16258 The specification allows use of one of two crypto suites. Crypto Suite 1 corresponds to the version offering ‘80-bit’  
16259 symmetric equivalent security, Crypto Suite 2 allows ‘128-bit’ symmetric equivalent security. Each utilizes different  
16260 key lengths and signature sizes and thus requires unique tags with different sizes.

## 16261 11.4.6 ECDSA Signature Sub-element (Crypto Suite 1)

16262 The ECDSA Signature sub-element contains a signature for the entire file as means of insuring that the data was not  
16263 modified at any point during its transmission from the signing device.

16264 If an image contains an ECDSA Signature Sub-element it SHALL be the last sub-element in the file.

16265 **Figure 11-4. ECDSA Signature**

Octets	2-bytes	4-bytes	8-bytes	42-bytes
<b>Data</b>	Tag ID: 0x0001	Length Field: 0x00000032	Signer IEEE Address	Signature Data

### 16266 11.4.6.1 Signer IEEE Address

16267 This field SHALL contain the IEEE address of the device that created the signature, in little endian format.

### 16268 11.4.6.2 Signature Data

16269 This field SHALL contain the ECDSA signature data, and is generated as described in the section 11.7.

## 16270 11.4.7 ECDSA Signing Certificate Sub-element

16271 This sub-element is used to include information about the authority that generated the signature for the OTA file.

16272

**Figure 11-5. ECDSA Signing Certificate Sub-element**

Octets	2-bytes	4-bytes	48-bytes
<b>Data</b>	Tag ID: 0x0002	Length Field: 0x00000030	ECDSA Certificate

16273 **11.4.7.1 ECDSA Certificate (Crypto Suite 1)**

16274 This SHALL contain the data for the ECDSA certificate of the device. The certificate SHALL be formatted as  
 16275 described in [Z9] in section C.4.2.2.4.

16276 **11.4.8 Image Integrity Code Sub-element**

16277 This sub-element includes a hash value used to verify the integrity of the OTA file.

16278

**Figure 11-6. Hash Value Sub-element**

Octets	2-bytes	4-bytes	16-bytes
<b>Data</b>	Tag ID: 0x0003	Length Field: 0x00000010	Hash Value

16279 **11.4.8.1 Hash Value**

16280 This hash value used to verify the integrity of the OTA file and the detail to generate the hash is listed in section  
 16281 11.7.2.

16282 **11.4.9 ECDSA Signature Sub-element (Crypto Suite 2)**

16283 The ECDSA Signature sub-element contains a signature for the entire file as means of insuring that the data was not  
 16284 modified at any point during its transmission from the signing device.

16285 If an image contains an ECDSA Signature Sub-element it SHALL be the last sub-element in the file.

16286

**Figure 11-7. ECDSA Signature**

Octets	2-bytes	4-bytes	8-bytes	72-bytes
<b>Data</b>	Tag ID: 0x0005	Length Field: 0x00000050	Signer IEEE Address	Signature Data

16287 **11.4.9.1 Signer IEEE Address**

16288 This field SHALL contain the IEEE address of the device that created the signature, in little endian format.

16289 **11.4.9.2 Signature Data**

16290 This field SHALL contain the ECDSA signature data, and is generated as described in the section 11.7.

16291 **11.4.10 ECDSA Signing Certificate Sub-element (Crypto Suite 2)**  
16292

16293 This sub-element is used to include information about the authority that generated the signature for the OTA file.

16294 **Figure 11-8. ECDSA Signing Certificate Sub-element**

Octets	2-bytes	4-bytes	74-bytes
Data	Tag ID: 0x0006	Length Field: 0x0000004A	ECDSA Certificate

16295 **11.4.10.1 ECDSA Certificate (Crypto Suite 2)**

16296 This SHALL contain the data for the ECDSA certificate of the device. The certificate SHALL be formatted as  
16297 described in [Z9] in section C.4.2.3.3.

16298 **11.5 OTA File Naming**

16299 OTA Upgrade cluster provides recommendation below regarding OTA Upgrade image file naming convention and  
16300 extension. This is an effort to assist the upgrade server in sorting different image files received from different  
16301 manufacturers.

16302 The OTA Upgrade image file name SHOULD contain the following information at the beginning of the name with  
16303 each field separated by a dash (“-”): manufacturer code, image type and file version. The value of each field stated  
16304 SHOULD be in hexadecimal number and in capital letter. Each manufacturer MAY append more information to the  
16305 name as seen fit to make the name more specific. The OTA Upgrade file extension SHOULD be “.zigbee”.

16306 An example of OTA Upgrade image file name and extension is “1001-00AB-10053519-upgradeMe.zigbee”.

16307 **11.6 Signatures**

16308 It is up to the application standard to determine whether or not a signature is necessary for over the air upgrade files.  
16309 If a standard has mandated the use of signatures then a device adhering to that standard SHALL only accept images  
16310 that have a signature sub-element. If such a device receives an OTA file that does not contain a signature sub-element,  
16311 then the device will discard the image and proceed with any further processing required by the specific application  
16312 standard. The device must verify the signature as described in the following sections prior to acting on any data inside  
16313 the file.

16314 If a standard does not require the use of signatures then devices MAY still choose to use images with signatures.  
16315 However, it is highly recommended that such a device only accept images either with signatures or without, but not  
16316 accept both. A device greatly reduces its security if it will accept signed or unsigned upgrade files.

16317 **11.7 ECDSA Signature Calculation**

16318 It is EXPECTED that in most all cases the signer device is not a real ZigBee device and is not part of any ZigBee  
16319 network. Therefore, the signer’s IEEE is not a real ZigBee device address, but the address of a virtual device that  
16320 exists only to sign upgrade images for a manufacturer and or a set of products. Its address SHOULD be separate from  
16321 the block of device addresses produced by a manufacturer as certified ZigBee devices.

16322 The signature calculation SHALL be performed as follows:



- 16323 1. A valid OTA image SHALL have previously been created including all the necessary header fields, tags, and  
16324 their data, in the image.
- 16325 2. The signer shall select a crypto suite to use based on its own security policies.
- 16326 3. An ECDSA signer certificate tag sub-element SHALL be constructed, based on the selected crypto suite, and  
16327 SHALL contain the certificate of the signing device, appended to the image.
- 16328 4. An ECDSA signature tag sub-element SHALL be constructed, based on the previously selected crypto suite,  
16329 including only the tag ID, the length of the tag, and the signer's IEEE address (see 11.4.6 and 11.4.9). No  
16330 actual signature data SHALL be included yet. The tag SHALL be appended to the image.
- 16331 5. The OTA image header SHALL be updated with a new total image size, including the signature certificate  
16332 tag sub-element that was added, and the full size of the ECDSA signature tag sub-element (see 11.4.6 and  
16333 11.4.9).
- 16334 6. A message digest SHALL be calculated over the entire image.
  - 16335 a. The message digest SHALL be computed by using the Matyas-Meyer-Oseas cryptographic hash  
16336 specified in the ZigBee core specification 05-3474-20 Section B.6. This uses the extended AES-  
16337 MMO hash proposed as a change to an earlier version of the ZigBee core specification 05-3474-20  
16338 Section B.6.
- 16339 2. The ECDSA algorithm SHALL be used to calculate the signature using the message digest and the signer  
16340 device's private key.
- 16341 3. The r and s components of the signature SHALL both be appended to the image. The r component SHALL  
16342 be appended first, and then the s component.

### 16343 **11.7.1 ECDSA Signature Verification**

16344 The signature of a completely downloaded OTA file SHALL be verified as follows.

- 16345 1. The ZigBee device SHALL first determine if the signer of the image is an authorized signer.
  - 16346 a. It does this by extracting the signer IEEE from ECDSA signature tag sub-element.
    - 16347 a. If an ECDSA signature tag sub-element is not found in the image, then the image SHALL be  
16348 discarded as invalid and no further processing SHALL be done.
    - 16349 b. The device SHALL compare the extracted signer IEEE with the list of local, known, authorized signers and  
16350 determine if there is a match.
    - 16351 c. If no match is found, then the image SHALL be discarded as invalid and no further processing SHALL be  
16352 done.
  - 16353 2. The device SHALL then check the ECDSA Crypto Suite of the image.
    - 16354 a. The device SHALL check which crypto suite is used for both the ECDSA Signature tag and ECDSA  
16355 Signing Certificate tag. If both elements do not use the same crypto suite, then the device SHALL discard  
16356 the image as invalid and no further processing SHALL be done.
    - 16357 b. The device SHALL check which crypto suites are locally allowed and supported. If the device does not  
16358 locally support the crypto suite used by the ECDSA Signing certificate tag or a security policy does not  
16359 allow its use for verifying locally, then it SHALL discard the image as invalid and no further processing  
16360 SHALL be done.
  - 16361 3. The device SHALL then obtain the certificate associated with the signer IEEE.
    - 16362 a. The device SHALL extract the signer certificate data from the ECDSA signing certificate sub-element.  
16363 If there is no ECDSA signing certificate tag sub-element, then it SHALL discard the image as invalid and  
16364 no further processing SHALL be done.

- 16365        b. The device SHALL verify that the signer IEEE address within the ECDSA signature tag sub-element  
16366        matches the subject field of the ECDSA signing certificate sub-element.
- 16367        **Note:** The subject field IEEE is in big-endian format and the signer IEEE is in little endian format.
- 16368        c. If the addresses do not match, then the image SHALL be discarded as invalid and no further processing  
16369        SHALL be done.
- 16370        4. The device SHALL then obtain the CA public key associated with the signer.
- 16371        a. The device SHALL obtain the IEEE of the CA public key from the issuer field within the ECDSA  
16372        certificate data of the ECDSA signing certificate sub-element.
- 16373        b. If the IEEE of the CA does not match its list of known CAs, or the public key for that CA could not be  
16374        locally obtained, then the image SHALL be discarded as invalid and no further processing SHALL be  
16375        done.
- 16376        5. The device SHALL then calculate the message digest of the image.
- 16377        The digest SHALL be calculated using the Matyas-Meyer-Oseas cryptographic hash function over the entire  
16378        image except for the signature data of the ECDSA signature sub-element.
- 16379        **Note:** The calculation SHALL include the signature tag ID of the ECDSA signature sub-element, the length  
16380        field of the ECDSA signature sub-element, and the signer IEEE field of the ECDSA signature sub-element.
- 16381        6. The signer’s public key SHALL be obtained by extracting it from the signer certificate.
- 16382        7. The device SHALL then pass the calculated digest value, signer certificate, and CA public key to the ECDSA  
16383        verification algorithm.
- 16384        8. If the ECDSA algorithm returns success, then the image SHALL be considered valid.
- 16385        9. If the ECDSA algorithm returns any other result, then the image SHALL be discarded as invalid and no further  
16386        processing SHALL be done.

## 16387        11.7.2 Image Integrity Code

16388        It is up to the application standard to determine whether or not an image integrity code is necessary for over the air  
16389        upgrade files. Standards that require the use of digital signatures SHALL NOT use this Image Integrity Hash Code  
16390        sub-element in conjunction to the ECDSA signature. If a standard has mandated the use of hash values then a device  
16391        adhering to that standard SHALL only accept images that have a valid hash sub-element. If such a device receives an  
16392        OTA file that does not contain a hash sub-element, then the device will discard the image and proceed with any further  
16393        processing required by the specific application standard. The device must verify the hash as described in the following  
16394        sections prior to acting on any data inside the file.

16395        The hash value provides protection against unintended data corruption. An OTA image which is hosted at a back-end  
16396        image repository might be stored and forwarded at several intermediate locations before it reaches the OTA server,  
16397        where it is typically stored on a local file system. There is a potential for this file being corrupted either during transfer  
16398        or as a result of file system errors. The hash value provides an interoperable way for OTA servers to detect corrupt  
16399        images before advertising such files to OTA clients. Otherwise corrupt images might only be detected by OTA clients  
16400        after complete download over-the-air. Since this condition cannot be detected by the OTA server, it would offer the  
16401        same (corrupt) file over and over again.

16402        If the application standard does not mandate the use of this hash value, it is strongly recommended that image integrity  
16403        is ascertained using another approach, for example a hash value (SHA-256 or comparable) that is maintained out-of-  
16404        band and provided by the device vendor together with the OTA image.

### 16405        11.7.2.1 Hash Value Calculation

16406        The hash value calculation SHALL be performed as follows:

- 16407 1. A valid OTA image SHALL have previously been created including all the necessary header fields, tags, and  
16408 their data, in the image.
- 16409 2. An AES-MMO hash value tag sub-element header SHALL be constructed including only the tag ID and the  
16410 length of the sub-element data (16 bytes). No actual data SHALL be included yet. The tag header SHALL be  
16411 appended to the image.
- 16412 3. The OTA image header SHALL be updated with a new total image size, including the hash tag sub-element  
16413 header that was added, and the full size of the hash tag sub-element ( $6 + 16 = 22$  bytes).
- 16414 4. The hash value SHALL be calculated using the Matyas-Meyer-Oseas cryptographic hash specified in section  
16415 B.6 of [R5]. The hash is calculated starting with the OTA image header and spanning just before the hash sub-  
16416 element header, i.e. the calculation takes in to account the first byte of the image header up to the last byte of  
16417 the sub-element preceding the hash sub-element.
- 16418 5. The computed hash value SHALL be appended to the image.

### 16419 **11.7.2.2 Hash Value Verification**

16420 The hash value of a complete OTA file SHALL be verified as follows.

- 16421 1. The device SHALL calculate the hash value of the image using the Matyas-Meyer-Oseas cryptographic hash  
16422 function. The hash is calculated starting with the OTA image header and spanning just before the hash sub-  
16423 element header, i.e. the calculation takes in to account the first byte of the image header up to the last byte of  
16424 the sub-element preceding the hash sub-element.
- 16425 2. The device SHALL then compare the calculated hash value with the data stored in the hash tag sub-element  
16426 data. If both octet strings are equal, the image SHALL be considered intact, otherwise the image SHALL be  
16427 considered corrupt.
- 16428 3. If the image is regarded corrupt, it SHALL be discarded as invalid and no further processing SHALL be done.

## 16429 **11.8 Discovery of the Upgrade Server**

16430 Before becoming part of the network, a device MAY be preprogrammed with the IEEE address of the authorized  
16431 upgrade server. In this case, once the device is part of the network, it SHALL discover the network address of the  
16432 upgrade server via ZDO network address discovery command.

16433 If the device is not preprogrammed with the upgrade server's IEEE address, the device SHALL discover the upgrade  
16434 server before it participates in any upgrade process. The device SHALL send Match Descriptor Request (ZDO  
16435 command) to discover an upgrade server by specifying a single OTA cluster ID in the input Cluster attribute. If the  
16436 receiving node is an upgrade server, it SHALL reply with Match Descriptor Response, with the (active) endpoint that  
16437 the OTA cluster is implemented on, hence, identifying itself as acting as server in OTA Upgrade cluster. Since Match  
16438 descriptor request MAY be sent as unicast or broadcast, the client MAY get multiple responses if there are more than  
16439 one server in the network. The client SHALL use the first response received. Each application standard SHOULD  
16440 specify the frequency of OTA server discovery done by the client. After discovering the OTA server's short ID via  
16441 the ZDO Match descriptor request the client SHALL discover the IEEE address of the upgrade server via ZDO IEEE  
16442 address discovery command and store the value in UpgradeServerID attribute.

16443 A node SHALL have an application link key with the Upgrade server; it SHALL request one prior to any OTA  
16444 operations.

16445 If the upgrade server is the trust center, it SHOULD use its trust center link key.

16446 In the case of the ZigBee Smart Energy standard, where the upgrade server is not the trust center, the device SHALL  
16447 perform partner link key request.

## 11.8.1 Server and Client

16448  
16449 The server must be able to store one or more OTA upgrade image(s). The server MAY notify devices in the network  
16450 when it receives new OTA upgrade image by sending an Image Notify Command. The Image Notify Command will  
16451 be received reliably only on ZR devices since ZED devices MAY have their radio off at the time. The Image Notify  
16452 Command MAY be sent as unicast or broadcast. If sent as broadcast, the message also has a jitter mechanism built in  
16453 to avoid the server being overwhelmed by the requests from the clients. If sent as unicast, the client SHALL ignore  
16454 the jitter value.

16455 The client device will send Query Next Image Request Command if the information in the Image Notify Command is  
16456 of interest and after applying the jitter value. All devices SHALL send in a Query Next Image Request Command  
16457 periodically regardless of whether an Image Notify was sent by the OTA server.

16458 When the device has received a response to its query indicating a new OTA upgrade image is available, the client  
16459 device SHALL request blocks of the OTA upgrade image. The process continues until the client receives all image  
16460 data. At that point, the client SHALL verify the integrity of the whole image received and send Upgrade End Request  
16461 Command along with the upgrade status. The server SHALL notify the client of when to upgrade to new image in the  
16462 Upgrade End Response.

16463 It is the responsibility of the server to ensure that all clients in the network are upgraded. The server MAY be told  
16464 which client to upgrade or it MAY keep a database of all clients in the network and track which client has not yet been  
16465 upgraded.

## 11.8.2 Sleepy Devices

16466  
16467 The upgrade server has no reliable way to immediately notify the sleepy devices of the availability of new OTA  
16468 Upgrade image, hence, the devices SHALL query the server periodically to learn if there are new images available.  
16469 The query for new upgrade image MAY be done as a separate event or it MAY be done in addition to normal scheduled  
16470 communication between the device and the server. The frequency as to how often the sleepy devices query the server  
16471 SHALL be specified by each application standard. Moreover, it is important to realize that the frequency that the  
16472 sleepy device checks for new image (sending Query Next Image command) determines how often the particular node  
16473 could be upgraded. This rate will also drive how fast code updates MAY be pushed out to each network. For the SE  
16474 1.x to SE 2.0 transition, if sleepy devices only check in once a month for the new image then it will likely to take over  
16475 a month to complete the transition. If the application standard fails to set any requirement on the sleepy device  
16476 checking for new images, then it is unlikely that the OTA upgrade feature will work reliably for those devices.

16477 It is a recommendation that sleepy devices SHALL make their best effort to poll more rapidly during the OTA Upgrade  
16478 Image download process in order to ensure that the download completes in a timely manner. However, it is  
16479 acknowledged that some sleepy devices MAY not be able to do so due to limitation on their batteries or due to other  
16480 reasons such as battery-less/Green Power devices. Hence, such devices MAY take much longer to complete the  
16481 download process.

## 11.9 Dependencies

16482  
16483 Each device that wishes to implement the OTA Upgrade cluster SHALL have the following:

- 16484 • ZigBee Device Object (ZDO) match descriptor request and response commands. The command is used to  
16485 discover upgrade server.
- 16486 • ZigBee Cluster Library (ZCL) global commands and basic cluster attributes.
- 16487 • Application Bootloader: To actually upgrade existing image with newly installed one on the additional memory  
16488 space. The implementation of the Bootloader along with its specification, for example, where it lives and its size  
16489 are outside the scope of this document.
- 16490 • Additional Memory Space SHALL be large enough to hold the whole OTA Upgrade Image: It is important to  
16491 be able to store the new image until the device receives a signal from the server to switch to running the image.

16492 This is because it MAY be necessary for all devices in the network to switch their images at once if the new  
 16493 image is not OTA compatible with the old one.

16494 • In addition, if the client device is composed of multiple processors; each requires separate image, then the  
 16495 additional memory space SHALL be large enough to hold all the images for all the processors that make up the  
 16496 device. In case of server devices, its additional memory space will depend on how many images the devices are  
 16497 planning to hold.

16498 • The specification of the additional memory space and its connection to the processor is outside the scope of this  
 16499 document.

## 16500 11.10 OTA Cluster Attributes

16501 Below are attributes defined for OTA Upgrade cluster. Currently, **all attributes are client side attributes** (only stored  
 16502 on the client). There is no server side attribute at the moment. All attributes with the exception of UpgradeServerID  
 16503 SHOULD be initialized to their default values before being used.

16504 **Table 11-10. Attributes of OTA Upgrade Cluster**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>UpgradeServerID</i>	EUI64	-	R	0xffffffffffffff	M
0x0001	<i>FileOffset</i>	uint32	<i>all</i>	R	0xffffffff	O
0x0002	<i>CurrentFileVersion</i>	uint32	<i>all</i>	R	0xffffffff	O
0x0003	<i>CurrentZigBeeStackVersion</i>	uint16	<i>all</i>	R	0xffff	O
0x0004	<i>DownloadedFileVersion</i>	uint32	<i>all</i>	R	0xffffffff	O
0x0005	<i>DownloadedZigBeeStackVersion</i>	uint16	<i>all</i>	R	0xffff	O
0x0006	<i>ImageUpgradeStatus</i>	enum8	<i>all</i>	R	0x00	M
0x0007	<i>Manufacturer ID</i>	uint16	<i>all</i>	R	-	O
0x0008	<i>Image Type ID</i>	uint16	<i>all</i>	R	-	O
0x0009	<i>MinimumBlockPeriod</i>	uint16	0x0000-0xfffe <sup>127</sup>	R	0 <sup>128</sup>	O
0x000a	<i>Image Stamp</i>	uint32	<i>all</i>	R		O
0x000b	<i>UpgradeActivationPolicy</i>	enum8	0x00-0x01	R	0x00	O
0x000c	<i>UpgradeTimeout Policy</i>	enum8	0x00-0x01	R	0x00	O

<sup>127</sup> CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

<sup>128</sup> CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

### 16505 **11.10.1 UpgradeServerID Attribute**

16506 The attribute is used to store the IEEE address of the upgrade server resulted from the discovery of the upgrade server's  
16507 identity. If the value is set to a non-zero value and corresponds to an IEEE address of a device that is no longer  
16508 accessible, a device MAY choose to discover a new Upgrade Server depending on its own security policies.

16509 The attribute is mandatory because it serves as a placeholder in a case where the client is programmed, during  
16510 manufacturing time, its upgrade server ID. In addition, the attribute is used to identify the current upgrade server the  
16511 client is using in a case where there are multiple upgrade servers in the network. The attribute is also helpful in a case  
16512 when a client has temporarily lost connection to the network (for example, via a reset or a rejoin), it SHALL try to  
16513 rediscover the upgrade server via network address discovery using the IEEE address stored in the attribute.

16514 By default, the value is 0xffffffffffff, which is an invalid IEEE address. The attribute is a client-side attribute and  
16515 stored on the client. Please refer to section 11.8 for a description of OTA server discovery.

### 16516 **11.10.2 FileOffset Attribute**

16517 The parameter indicates the current location in the OTA upgrade image. It is essentially the (start of the) address of  
16518 the image data that is being transferred from the OTA server to the client. The attribute is optional on the client and is  
16519 made available in a case where the server wants to track the upgrade process of a particular client.

### 16520 **11.10.3 CurrentFileVersion Attribute**

16521 The file version of the running firmware image on the device. The information is available for the server to query via  
16522 ZCL read attribute command. The attribute is optional on the client.

### 16523 **11.10.4 CurrentZigBeeStackVersion Attribute**

16524 The ZigBee stack version of the running image on the device. The information is available for the server to query via  
16525 ZCL read attribute command. The attribute is optional on the client. See 11.4.2.8 for values.

### 16526 **11.10.5 DownloadedFileVersion Attribute**

16527 The file version of the downloaded image on additional memory space on the device. The information is available for  
16528 the server to query via ZCL read attribute command. The information is useful for the OTA upgrade management, so  
16529 the server MAY ensure that each client has downloaded the correct file version before initiate the upgrade. The  
16530 attribute is optional on the client.

### 16531 **11.10.6 DownloadedZigBeeStackVersion Attribute**

16532 The ZigBee stack version of the downloaded image on additional memory space on the device. The information is  
16533 available for the server to query via ZCL read attribute command. The information is useful for the OTA upgrade  
16534 management, so the server SHALL ensure that each client has downloaded the correct ZigBee stack version before  
16535 initiate the upgrade. The attribute is optional on the client.

### 16536 **11.10.7 ImageUpgradeStatus Attribute**

16537 The upgrade status of the client device. The status indicates where the client device is at in terms of the download and  
16538 upgrade process. The status helps to indicate whether the client has completed the download process and whether it is  
16539 ready to upgrade to the new image. The status MAY be queried by the server via ZCL read attribute command. Hence,  
16540 the server MAY not be able to reliably query the status of ZED client since the device MAY have its radio off.

16541

**Table 11-11. Image Upgrade Status Attribute Values**

Image Upgrade Status Values	Description
0x00	Normal
0x01	Download in progress
0x02	Download complete
0x03	Waiting to upgrade
0x04	Count down
0x05	Wait for more
0x06	Waiting to Upgrade via External Event

16542

16543 Normal status typically means the device has not participated in any download process. Additionally, the client  
 16544 SHALL set its upgrade status back to Normal if the previous upgrade process was not successful.

16545 Download in progress status is used from when the client device receives SUCCESS status in the Query Next Image  
 16546 Response command from the server prior to when the device receives all the image data it needs.

16547 Download complete status indicates the client has received all data blocks required and it has already verified the OTA  
 16548 Upgrade Image signature (if applied) and has already written the image onto its additional memory space. The status  
 16549 will be modified as soon as the client receives Upgrade End Response command from the server.

16550 Wait to upgrade status indicates that the client is told by the server to wait until another (upgrade) command is sent  
 16551 from the server to indicate the client to upgrade its image.

16552 Count down status indicates that the server has notified the client to count down to when it SHALL upgrade its image.

16553 Wait for more (upgrade) image indicates that the client is still waiting to receive more OTA upgrade image files from  
 16554 the server. This is true for a client device that is composed of multiple processors and each processor requires different  
 16555 image. The client SHALL be in this state until it has received all necessary OTA upgrade images, then it SHALL  
 16556 transition to Download complete state.

## 16557 **11.10.8 Manufacturer ID Attribute**

16558 This attribute SHALL reflect the ZigBee assigned value for the manufacturer of the device. See also section 11.4.2.5.

## 16559 **11.10.9 Image Type ID Attribute**

16560 This attribute SHALL indicate the image type identifier of the file that the client is currently downloading, or a file  
 16561 that has been completely downloaded but not upgraded to yet. The value of this attribute SHALL be 0xFFFF when  
 16562 the client is not downloading a file or is not waiting to apply an upgrade.

## 16563 **11.10.10 MinimumBlockPeriod Attribute**

16564 This attribute acts as a rate limiting feature for the server to slow down the client download and prevent saturating the  
 16565 network with block requests. The attribute lives on the client but can be changed during a download if rate limiting is  
 16566 supported by both devices.

16567 This attribute SHALL reflect the minimum delay between Image Block Request commands generated by the client in  
16568 milliseconds<sup>129</sup>. The value of this attribute SHALL be updated when the rate is changed by the server, but SHOULD  
16569 reflect the client default when an upgrade is not in progress or a server does not support this feature.

### 16570 11.10.11 Image Stamp Attribute

16571 This attribute acts as a second verification to identify the image in the case that sometimes developers of the application  
16572 have forgotten to increase the firmware version attribute. It is a 32 bit value and has a valid range from 0x00000000  
16573 to 0xFFFFFFFF. This attribute value must be consistent during the lifetime of the same image and also must be unique  
16574 for each different build of the image. This attribute value SHOULD not be hardcoded or generated by any manual  
16575 process. This attribute value SHOULD be generated by performing a hash or checksum on the entire image. There are  
16576 two possible methods to generate this checksum. It can be generated dynamically during runtime of the application or  
16577 it can be generated during compile time of the application.

### 16578 11.10.12 UpgradeActivationPolicy Attribute

16579 This attribute indicates what behavior the client device supports for activating a fully downloaded but not installed  
16580 upgrade image. Table 11-12 below lists the enumerated values and the descriptions.

16581 **Table 11-12. UpgradeActivationPolicy Enumerations**

Policy Enumeration Value	Short Name	Description
0x00	OTA Server Activation Allowed	This value indicates that the OTA server's command, to tell the device when to upgrade, will be applied by the client.
0x01	Out-of-band Activation Only	This value indicates that the activation of the image is done via out-of-band mechanisms. Attempts by the OTA server to tell the client to install the image will be rejected. Examples of an out-of-band mechanism to apply the image are: user prompt, or non-ZigBee protocol message.

16582  
16583 Client devices with an *UpgradeActivationPolicy* value of 0x01 SHALL still send an *UpgradeEndRequest* command  
16584 to the OTA Server at the completion of their download. In this case, clients SHALL NOT process an  
16585 *UpgradeEndResponse* with a status of SUCCESS unless it has an *UpgradeTime* of 0xFFFFFFFF; upon receipt of an  
16586 *UpgradeEndResponse* with a status of SUCCESS, but having an *UpgradeTime* field other than 0xFFFFFFFF, the  
16587 client SHALL send back a Default Response with a status of NOT\_AUTHORIZED.

16588 In the absence of this optional attribute, the default value of 0x00 shall be assumed.

### 16589 11.10.13 UpgradeTimeoutPolicy Attribute

16590 This attribute indicates what behavior the client device supports for activating a fully downloaded image when the  
16591 OTA server cannot be reached.

16592 There may be circumstances when the OTA client is waiting on an explicit activation command and yet the activation  
16593 command cannot be retrieved. This may be due to the fact that the OTA server is down, or, if *UpgradeActivationPolicy*  
16594 is 0x01, the out-of-band communications mechanism is inaccessible.

<sup>129</sup> CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)



16595 In these circumstances the behavior of the device is dictated by the UpgradeTimeoutPolicy. After enough failed  
 16596 attempts to retrieve the activation command without any response, the OTA client’s behavior SHALL be dictated by  
 16597 Table 11-13.

16598 When the UpgradeTimeoutPolicy attribute is set to 0x00 and the UpgradeActivationPolicy is 0x00, section 11.16  
 16599 defines the requirements on how often retries are performed and at what required intervals. If the  
 16600 UpgradeTimeoutPolicy attribute is set to 0x00 and the UpgradeActivationPolicy is 0x01, any retry mechanism and  
 16601 timeouts are manufacturer specific. Whilst there may be situations where the mechanisms defined in section 11.16  
 16602 could be disabled when the UpgradeActivationPolicy is 0x00, the setting of the UpgradeTimeoutPolicy attribute to  
 16603 0x01 in this case is currently reserved.

16604 In the absence of this optional attribute, the default value of 0x00 shall be assumed.

16605 **Table 11-13. UpgradeTimeoutPolicy Enumerations**

Policy Enumeration Value	Short Name	Description
0x00	Apply Upgrade After Timeout	After the specified time has elapsed and the number of required retry attempts has been made, the device SHALL apply a downloaded but not installed image.
0x01	Do not Apply Upgrade After Timeout	No amount of time or failed attempts to retrieve the activation command SHALL trigger the device to apply a downloaded but not installed image.

16606

## 16607 11.11 OTA Cluster Parameters

16608 Below are defined parameters for OTA Upgrade cluster server. These values are considered as parameters and not  
 16609 attributes because their values tend to change often and are not static. Moreover, some of the parameters MAY have  
 16610 multiple values on the upgrade server at one instance. For example, for DataSize parameter, the value MAY be  
 16611 different for each OTA upgrade process. These parameters are included in commands sent from server to client. The  
 16612 parameters cannot be read or written via ZCL global commands.

16613 **Table 11-14. Parameters of OTA Upgrade Cluster**

Name	Type	Range	Default	M/O
QueryJitter	uint8	0x01 – 0x64	0x32	M
DataSize	uint8	0x00 – 0xff	0xff	M
OTAImageData	Opaque	Varied	all 0xff’s	M
CurrentTime	UTC <sup>130</sup>	all	0xffffffff	M
UpgradeTime or RequestTime	UTC <sup>131</sup>	all	0xffffffff	M

<sup>130</sup> CCB 2228 UTC time, not unsigned 32-bit integer

<sup>131</sup> CCB 2228 UTC time, not unsigned 32-bit integer

### 11.11.1 QueryJitter Parameter

16615 The parameter is part of Image Notify Command sent by the upgrade server. The parameter indicates whether the  
16616 client receiving Image Notify Command SHOULD send in Query Next Image Request command or not.

16617 The server chooses the parameter value between 1 and 100 (inclusively) and includes it in the Image Notify Command.  
16618 On receipt of the command, the client will examine other information (the manufacturer code and image type) to  
16619 determine if they match its own values. If they do not, it SHALL discard the command and no further processing  
16620 SHALL continue. If they do match, then it will determine whether or not it SHOULD query the upgrade server. It  
16621 does this by randomly choosing a number between 1 and 100 and comparing it to the value of the QueryJitter parameter  
16622 received. If it is less than or equal to the QueryJitter value from the server, it SHALL continue with the query process.  
16623 If not, then it SHALL discard the command and no further processing SHALL continue.

16624 By using the QueryJitter parameter, it prevents a single notification of a new OTA upgrade image from flooding the  
16625 upgrade server with requests from clients.

### 11.11.2 DataSize Parameter

16627 A value that indicates the length of the OTA image data included in the (Image Block Response) command payload  
16628 sent from the server to client.

### 11.11.3 OTAImageData Parameter

16630 This is a part of OTA upgrade image being sent over the air. The length of the data is dictated by the data size  
16631 parameter. The server does not need to understand the meaning of the data, only the client does. The data MAY also  
16632 be compressed or encrypted to increase efficiency or security.

16633 The parameter is a series of octets and is used with the file offset value (defined in section 11.10.2) to indicate the  
16634 location of the data and the data size value to indicate the length of the data.

### 11.11.4 CurrentTime and UpgradeTime/RequestTime Parameters

16637 If CurrentTime and UpgradeTime are used in the command (ex. Upgrade End Response), the server uses the  
16638 parameters to notify the client when to upgrade to the new image. If CurrentTime and RequestTime are used in the  
16639 command (ex. Image Block Response), the server is notifying the client when to request for more upgrade data. The  
16640 CurrentTime indicates the current time of the OTA server. The UpgradeTime indicates the time that the client SHALL  
16641 upgrade to running new image. The RequestTime indicates when the client SHALL request for more data.

16642 The value of the parameters and their interpretation MAY be different depending on whether the devices support ZCL  
16643 Time cluster or not. If ZCL Time cluster is supported, the values of both parameters MAY indicate the UTC Time  
16644 values that represent the Universal Time Coordinated (UTC) time. If the device does not support ZCL Time cluster,  
16645 then it SHALL compute the offset time value from the difference between the two time parameters. The resulted offset  
16646 time is in seconds. A device that does support the time cluster MAY use offset time instead of UTC Time when it  
16647 sends messages that reference the time according to Table 11-15.

16648 The table below shows how to interpret the time parameter values depending on whether Time cluster is supported on  
16649 the device. The intention here is to be able to support a mixed network of nodes that MAY not all support Time cluster.

16650 **Table 11-15. Meaning of CurrentTime and UpgradeTime Parameters**

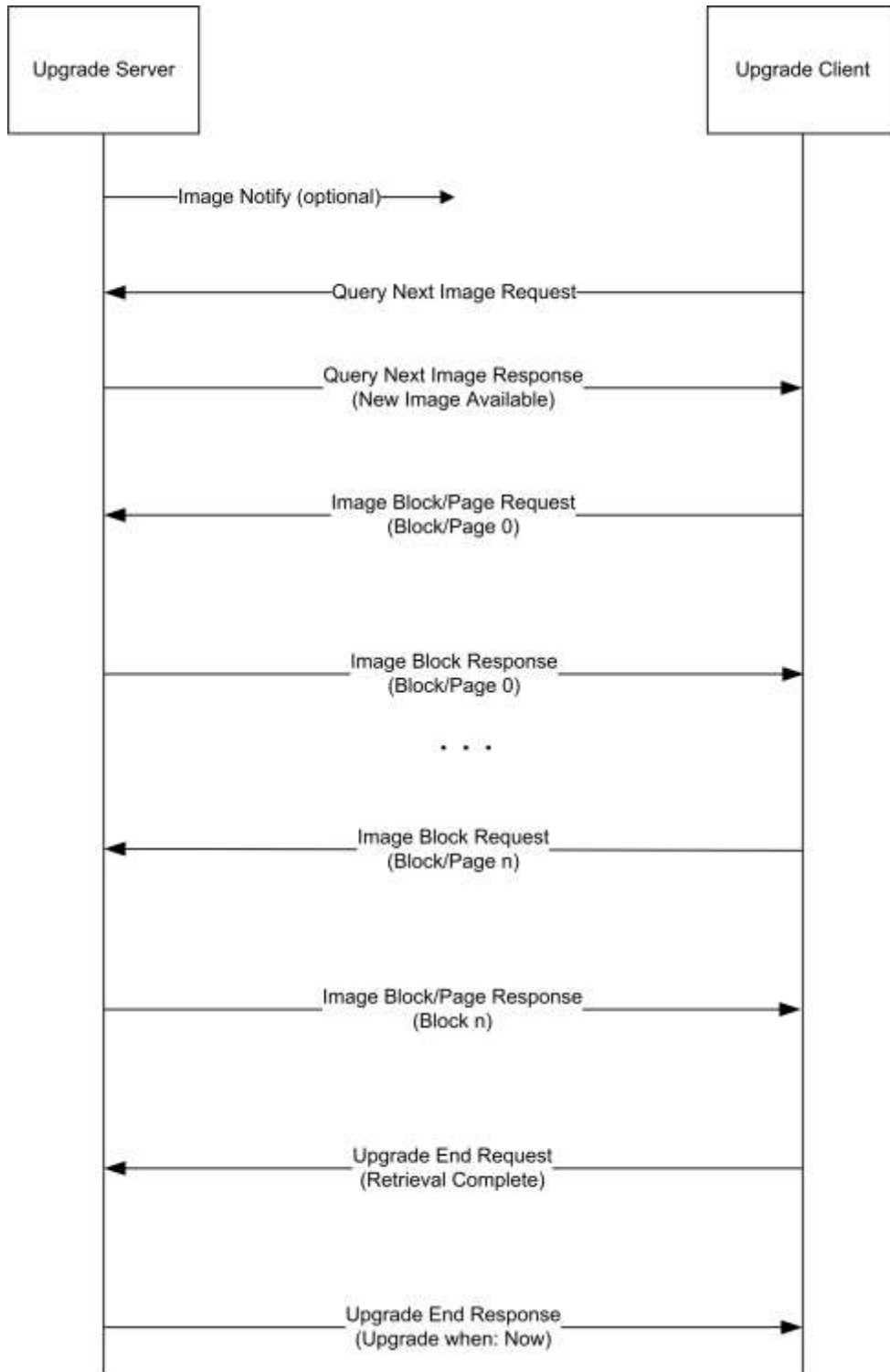
CurrentTime Value	UpgradeTime or RequestTime Value	Description
0x00000000	Any	Device SHALL use UpgradeTime or RequestTime as an offset time from now.

CurrentTime Value	UpgradeTime or RequestTime Value	Description
0x00000001 – 0xfffffffffe	Any	Server supports Time cluster; client SHALL use UpgradeTime/RequestTime value as UTCTime if it also supports Time cluster or it SHALL compute the offset time if it does not.
Any	0xffffffff	The client SHOULD wait for a (upgrade) command from the server. Note that value of 0xffffffff SHOULD not be used for RequestTime.

- 16651
- 16652 For client devices with an *UpgradeActivationPolicy* value of 0x00, using a value of all 0xFF's for UpgradeTime to
- 16653 indicate a wait (for an Upgrade End Response command from the server) on ZED client devices is not recommended
- 16654 since an upgrade server SHOULD not be assumed to know the wake up cycle of the end device; hence it is not
- 16655 guaranteed that the end device will receive the upgrade command. If the wait value (0xffffffff) is used on a ZED client,
- 16656 the client SHOULD keep querying the server at a reasonable rate (not faster than once every 60 minutes) to see if it is
- 16657 time to upgrade. Client devices with an *UpgradeActivationPolicy* value of 0x01 will expect an UpgradeTime of all
- 16658 0xFF's, but SHALL NOT keep querying the server.
- 16659 Using value of all 0xFF's for RequestTime to indicate an indefinite wait time is not recommended. If the server does
- 16660 not know when it will have the image data ready, it SHALL use a reasonable wait time and when the client resends
- 16661 the image request, the server SHALL keep telling it to wait. There is no limit to how many times the server SHOULD
- 16662 the client to wait for the upgrade image. Using value of 0xffffffff SHALL cause the client to wait indefinitely and
- 16663 server MAY not have a way to tell the client to stop waiting especially for ZED client.

## 11.12 OTA Upgrade Diagram

16665 **Figure 11-9. OTA Upgrade Message Diagram**



16666  
16667

16668 Please refer to section 11.13 for the command description used in Figure 11-9.

## 16669 **11.13 Command Frames**

16670 OTA upgrade messages do not differ from typical ZigBee APS messages so the upgrade process SHOULD not  
 16671 interrupt the general network operation. All OTA Upgrade cluster commands SHALL be sent with APS retry option,  
 16672 hence, require APS acknowledgement; unless stated otherwise.

16673 OTA Upgrade cluster commands, the frame control value SHALL follow the description below:

- 16674 • Frame type is 0x01: commands are cluster specific (not a global command).
- 16675 • Manufacturer specific is 0x00: commands are not manufacturer specific.
- 16676 • Direction: SHALL be either 0x00 (client->server) or 0x01 (server->client) depending on the commands.
- 16677 • Disable default response is 0x00 for all OTA request commands sent from client to server: default response  
 16678 command SHALL be sent when the server receives OTA Upgrade cluster request commands that it does not  
 16679 support or in case an error case happens. A detailed explanation of each error case along with its recommended  
 16680 action is described for each OTA cluster command.
- 16681 • Disable default response is 0x01 for all OTA response commands (sent from server to client) and for  
 16682 broadcast/multicast Image Notify command: default response command is not sent when the client receives a  
 16683 valid OTA Upgrade cluster response commands or when it receives broadcast or multicast Image Notify  
 16684 command. However, if a client receives invalid OTA Upgrade cluster response command, a default response  
 16685 SHALL be sent. A detailed explanation of each error case along with its recommended action is described for  
 16686 each OTA cluster command.

### 16687 **11.13.1 OTA Cluster Command Identifiers**

16688 Command identifier values are listed in Table 11-16 below.

16689 **Table 11-16. OTA Upgrade Cluster Command Frames**

<b>Id</b>	<b>Name</b>	<b>Direction</b>	<b>Disable Default Response</b>	<b>M/O</b>
0x00	<i>Image Notify</i>	Server -> Client(s) (0x01)	Set if sent as broadcast or multicast; Not Set if sent as unicast	O
0x01	<i>Query Next Image Request</i>	Client -> Server (0x00)	Not Set	M
0x02	<i>Query Next Image Response</i>	Server -> Client (0x01)	Set	M
0x03	<i>Image Block Request</i>	Client -> Server (0x00)	Not Set	M
0x04	<i>Image Page Request</i>	Client -> Server (0x00)	Not Set	O
0x05	<i>Image Block Response</i>	Server -> Client (0x01)	Set	M
0x06	<i>Upgrade End Request</i>	Client -> Server (0x00)	Not Set	M
0x07	<i>Upgrade End Response</i>	Server -> Client (0x01)	Set	M

<b>Id</b>	<b>Name</b>	<b>Direction</b>	<b>Disable Default Response</b>	<b>M/O</b>
0x08	<i>Query Device Specific File Request</i>	Client -> Server (0x00)	Not Set	O
0x09	<i>Query Device Specific File Response</i>	Server -> Client (0x01)	Set	O

### 11.13.2 OTA Cluster Status Codes

16690  
16691  
16692  
16693  
16694

OTA Upgrade cluster uses ZCL defined status codes during the upgrade process. These status codes are included as values in status field in payload of OTA Upgrade cluster's response commands and in default response command. Some of the status codes are new and are still in the CCB process in order to be included in the ZCL specification.

**Table 11-17. Status Code Defined and Used by OTA Upgrade Cluster**

<b>ZCL Status Code</b>	<b>Value</b>	<b>Description</b>
SUCCESS	0x00	Success Operation
ABORT	0x95	Failed case when a client or a server decides to abort the upgrade process.
NOT_AUTHORIZED	0x7E	Server is not authorized to upgrade the client
INVALID_IMAGE	0x96	Invalid OTA upgrade image (ex. failed signature validation or signer information check or CRC check)
WAIT_FOR_DATA	0x97	Server does not have data block available yet
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for a particular client
MALFORMED_COMMAND	0x80	The command received is badly formatted. It usually means the command is missing certain fields or values included in the fields are invalid ex. invalid jitter value, invalid payload type value, invalid time value, invalid data size value, invalid image type value, invalid manufacturer code value and invalid file offset value
UNSUP_CLUSTER_COMMAND	0x81	Such command is not supported on the device
REQUIRE_MORE_IMAGE	0x99	The client still requires more OTA upgrade image files in order to successfully upgrade

### 11.13.3 Image Notify Command

16695  
16696  
16697  
16698  
16699

The purpose of sending Image Notify command is so the server has a way to notify client devices of when the OTA upgrade images are available for them. It eliminates the need for ZR client devices having to check with the server periodically of when the new images are available. However, all client devices still need to send in Query Next Image Request command in order to officially start the OTA upgrade process.

16700 **11.13.3.1 Payload Format**

16701 **Figure 11-10. Format of Image Notify Command Payload**

Octets	1	1	0/2	0/2	0/4
<b>Data Type</b>	enum8	uint8	uint16	uint16	uint32
<b>Field Name</b>	Payload type	Query jitter	Manufacturer code	Image type	(new) File version

16702 **11.13.3.2 Payload Field Definitions**

16703 **11.13.3.2.1 Image Notify Command Payload Type**

16704 **Table 11-18. Image Notify Command Payload Type**

Payload Type Values	Description
0x00	Query jitter
0x01	Query jitter and manufacturer code
0x02	Query jitter, manufacturer code, and image type
0x03	Query jitter, manufacturer code, image type, and new file version

16705 **11.13.3.2.2 Query Jitter**

16706 See section 11.11.1 for detailed description.

16707 **11.13.3.2.3 Manufacturer Code**

16708 Manufacturer code when included in the command SHOULD contain the specific value that indicates certain  
 16709 manufacturer. If the server intends for the command to be applied to all manufacturers, then the value SHOULD be  
 16710 omitted. See section 2 for detailed description.

16711 **11.13.3.2.4 Image Type**

16712 Image type when included in the command SHOULD contain the specific value that indicates certain file type. If the  
 16713 server intends for the command to be applied to all image type values then the wild card value (0xffff) SHOULD be  
 16714 used. See section 11.4.2.6 for detailed description.

16715 **11.13.3.2.5 (new) File Version**

16716 The value SHALL be the OTA upgrade file version that the server tries to upgrade client devices in the network to. If  
 16717 the server intends for the command to be applied to all file version values then the wild card value (0xffffffff)  
 16718 SHOULD be used. See section 11.10.3 for detailed description.

### 16719 11.13.3.3 When Generated

16720 For ZR client devices, the upgrade server MAY send out a unicast, broadcast, or multicast indicating it has the next  
16721 upgrade image, via an Image Notify command. Since the command MAY not have APS security (if it is broadcast or  
16722 multicast), it is considered purely informational and *not authoritative*. Even in the case of a unicast, ZR SHALL  
16723 continue to perform the query process described in later section.

16724 When the command is sent with payload type value of zero, it generally means the server wishes to notify all clients  
16725 disregard of their manufacturers, image types or file versions. Query jitter is needed to protect the server from being  
16726 flooded with clients' queries for next image.

16727 The server MAY choose to send the Image Notify command to a more specific group of client devices by choosing  
16728 higher payload type value. Only devices with matching information as the ones included in the Image Notify command  
16729 will send back queries for next image.

16730 However, payload type value of 0x03 has a slightly different effect. If the client device has all the information  
16731 matching those included in the command including the new file version, the device SHALL then ignore the command.  
16732 This indicates that the device has already gone through the upgrade process. This is to prevent the device from  
16733 downloading the same image version multiple times. This is only true if the command is sent as broadcast/multicast.

16734 Query jitter value indicates how the server wants to space out the responses from the client; generally as a result of  
16735 sending the command as broadcast or multicast. The client will only respond back if it randomly picks a value that is  
16736 equal or smaller than the query jitter value. When sending Image Notify command as broadcast or multicast, the  
16737 Disable Default Response bit in ZCL header must be set (to 0x01) to avoid the client from sending any default response  
16738 back to the upgrade server. This agrees with section 2.4.12.

16739 If the command is sent as unicast, the payload type value MAY be zero and the Query jitter value MAY be the  
16740 maximum value of 100 to signal the client to send in Query Next Image Request. The server MAY choose to use other  
16741 payload type values besides zero when sending as unicast. However, since the server already knows the specific client  
16742 (address) it wants to upgrade so other information is generally irrelevant.

16743 The upgrade server MAY choose to send Image Notify command to avoid having ZR clients sending in Query Next  
16744 Image Request to it periodically.

### 16745 11.13.3.4 Effect on Receipt

16746 On receipt of a unicast Image Notify command, the device SHALL always send a Query Next Image request back to  
16747 the upgrade server.

16748 On receipt of a broadcast or multicast Image Notify command, the device SHALL keep examining each field included  
16749 in the payload with its own value. For each field, if the value matches its own, it SHALL proceed to examine the next  
16750 field. If values in all three fields (naming manufacturer code, image type and new file version) match its own values,  
16751 then it SHALL discard the command. The new file version in the payload SHALL be a match, it either matches the  
16752 device's current running file version or matches the downloaded file version (on the additional memory space).

16753 If manufacturer code or the image type values in the payload does not match the device's own value, it SHALL discard  
16754 the command. For payload type value of 0x01, if manufacturer code matches the device's own value, the device  
16755 SHALL proceed. For payload type value of 0x02, if both manufacturer code and image type match the device's own  
16756 values, the device SHALL proceed. For payload type value of 0x03, if both manufacturer code and image type match  
16757 the device's own values but the new file version is not a match, the device SHALL proceed. In this case, the (new)  
16758 file version MAY be lower or higher than the device's file version to indicate a downgrade or an upgrade of the  
16759 firmware respectively.

16760 To proceed, the device SHALL randomly choose a number between 1 and 100 and compare it to the value of the  
16761 QueryJitter value in the received message. If the generated value is less than or equal to the received value for  
16762 QueryJitter, it SHALL query the upgrade server. If not, then it SHALL discard the message and no further processing  
16763 SHALL continue.

16764 By using the QueryJitter field, a server MAY limit the number of devices that will query it for a new OTA upgrade  
16765 image, preventing a single notification of a new software image from flooding the upgrade server with requests.



16766 In application standards that mandate APS encryption for OTA upgrade cluster messages, OTA messages sent as  
 16767 broadcast or multicast SHOULD be dropped by the receivers.

### 16768 11.13.3.5 Handling Error Cases

16769 The section describes all possible error cases that the client MAY detect upon reception invalid Image Notify  
 16770 command from the server, along with the action that SHALL be taken.

16771 For invalid broadcast or multicast Image Notify command, for example, out-of-range query jitter value is used, or the  
 16772 reserved payload type value is used, or the command is badly formatted, the client SHALL ignore such command and  
 16773 no processing SHALL be done. In addition, the broadcast/multicast command SHALL have disable default response  
 16774 bit in the ZCL frame control set to 0x01.

16775 The cases below describe how to handle invalid Image Notify command that is sent as unicast. Such command SHALL  
 16776 not have default response bit set.

#### 16777 11.13.3.5.1 Malformed Command

16778 Scenarios for this error case include unicast Image Notify command with payload type of non-zero value, unicast  
 16779 Image Notify command with query jitter value that is not 100, broadcast Image Notify command with out of range  
 16780 query jitter value. In such scenario, the client SHOULD ignore the invalid message and SHALL send default response  
 16781 command with MALFORMED\_COMMAND status to the server.

## 16782 11.13.4 Query Next Image Request Command

### 16783 11.13.4.1 Payload Format

16784 **Figure 11-11. Format of Query Next Image Request Command Payload**

Octets	1	2	2	4	0/2
Data Type	map8 <sup>132</sup>	uint16	uint16	uint32	uint16
Field Name	Field control	Manufacturer code	Image type	(Current) File version	Hardware version

### 16785 11.13.4.2 Payload Field Definitions

#### 16786 11.13.4.2.1 Query Next Image Request Command Field Control

16787 The field control indicates whether additional information such as device’s current running hardware version is  
 16788 included as part of the Query Next Image Request command.

16789 **Table 11-19. Query Next Image Request Field Control Bitmask**

Bits	Name
0	Hardware Version Present

<sup>132</sup> CCB 2219 Field Control is a bitmap, not an unsigned integer

16790 **11.13.4.2.2 Manufacturer Code**

16791 The value SHALL be the device's assigned manufacturer code. Wild card value SHALL not be used in this case. See  
16792 Chapter 2 for detailed description.

16793 **11.13.4.2.3 Image Type**

16794 The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed  
16795 description. For other image type values, Query Device Specific File Request command SHOULD be used.

16796 **11.13.4.2.4 File Version (current)**

16797 The file version included in the payload represents the device's current running image version. Wild card value  
16798 SHALL not be used in this case. See section 11.10.3 for more detailed description.

16799 **11.13.4.2.5 Hardware Version (optional)**

16800 The hardware version if included in the payload represents the device's current running hardware version. Wild card  
16801 value SHALL not be used in this case. See section 11.4.2.13 for hardware version format description.

16802 **11.13.4.3 When Generated**

16803 Client devices SHALL send a Query Next Image Request command to the server to see if there is new OTA upgrade  
16804 image available. ZR devices MAY send the command after receiving Image Notify command. ZED device SHALL  
16805 periodically wake up and send the command to the upgrade server. Client devices query what the *next* image is, based  
16806 on their own information.

16807 **11.13.4.4 Effect on Receipt**

16808 The server takes the client's information in the command and determines whether it has a suitable image for the  
16809 particular client. The decision SHOULD be based on specific policy that is specific to the upgrade server and outside  
16810 the scope of this document... However, a recommended default policy is for the server to send back a response that  
16811 indicates the availability of an image that matches the manufacturer code, image type, and the highest available file  
16812 version of that image on the server. However, the server MAY choose to upgrade or downgrade a clients' image, as  
16813 its policy dictates. If client's hardware version is included in the command, the server SHALL examine the value  
16814 against the minimum and maximum hardware versions included in the OTA file header.

16815 How the server retrieves and stores the clients' file is also outside the scope of this document. The server MAY have  
16816 a backend communication to retrieve the images or it MAY have database software to manage file storage.

16817 **11.13.4.5 Handling Error Cases**

16818 All error cases resulting from receiving Query Next Image Request command are handled by the corresponding Query  
16819 Next Image Response command with the exception of the malformed request command described below that is  
16820 handled by default response command. Please refer to section 11.13.5.3 for more information regarding how the Query  
16821 Next Image response command is generated.

16822 **11.13.4.5.1 Malformed Command**

16823 Upon reception a badly formatted Query Next Image Request command, for example, the command is missing one of  
16824 the payload fields; the server SHALL send default response command with MALFORMED\_COMMAND status to  
16825 the client and it SHALL not process the command further.

16826 **11.13.5 Query Next Image Response Command**

16827 **11.13.5.1 Payload Format**

16828 **Figure 11-12. Format of Query Next Image Response Command Payload**

Octets	1	0/2	0/2	0/4	0/4
Data Type	enum8 <sup>133</sup>	uint16	uint16	uint32	uint32
Field Name	Status	Manufacturer code	Image type	File version	Image size

16829 **11.13.5.2 Payload Field Definitions**

16830 **11.13.5.2.1 Query Next Image Response Status**

16831 Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field SHALL  
 16832 be present. See section 11.13.2 for a complete list and description of OTA Cluster status codes.

16833 **11.13.5.2.2 Manufacturer Code**

16834 The value SHALL be the one received by the server in the Query Next Image Request command. See Chapter 2 for  
 16835 detailed description.

16836 **11.13.5.2.3 Image Type**

16837 The value SHALL be the one received by the server in the Query Next Image Request command. See section 11.4.2.6  
 16838 for detailed description.

16839 **11.13.5.2.4 File Version**

16840 The file version indicates the image version that the client is required to install. The version value MAY be lower than  
 16841 the current image version on the client if the server decides to perform a downgrade. The version value MAY not be  
 16842 the same as the client’s current version. Reinstallation of the same software version is not supported. In general, the  
 16843 version value SHOULD be higher than the current image version on the client to indicate an upgrade. See section  
 16844 11.4.2.7 for more description.

16845 **11.13.5.2.5 Image Size**

16846 The value represents the total size of the image (in bytes) including header and all sub-elements. See section 11.4.2.10  
 16847 for more description.

<sup>133</sup> CCB 2220 Status is an enumeration, not an unsigned integer

### 16848 **11.13.5.3 When Generated**

16849 The upgrade server sends a Query Next Image Response with one of the following status: SUCCESS,  
16850 NO\_IMAGE\_AVAILABLE or NOT\_AUTHORIZED. When a SUCCESS status is sent, it is considered to be the  
16851 explicit authorization to a device by the upgrade server that the device MAY upgrade to a specific software image.

16852 A status of NO\_IMAGE\_AVAILABLE indicates that the server is authorized to upgrade the client but it currently  
16853 does not have the (new) OTA upgrade image available for the client. For all clients (both ZR and ZED), they SHALL  
16854 continue sending Query Next Image Requests to the server periodically until an image becomes available.

16855 A status of NOT\_AUTHORIZED indicates the server is not authorized to upgrade the client. In this case, the client  
16856 MAY perform discovery again to find another upgrade server. The client MAY implement an intelligence to avoid  
16857 querying the same unauthorized server.

### 16858 **11.13.5.4 Effect on Receipt**

16859 A status of SUCCESS in the Query Next Image response indicates to the client that the server has a new  
16860 OTA upgrade image. If the file version contained in the Query Next Image Response is the same as the  
16861 CurrentFileVersion attribute (the current running version of software) *or the DownloadedFileVersion*  
16862 *attribute for the specified Image Type*<sup>134</sup>, then the message SHOULD<sup>135</sup> be discarded and no further  
16863 processing SHOULD<sup>136</sup> be done. Reinstallation of the same software version is not supported. Otherwise  
16864 the client MAY<sup>137</sup> begin requesting blocks of the image using the Image Block Request command. A ZED  
16865 client MAY choose to change its wake cycle to retrieve the image more quickly.

### 16866 **11.13.5.5 Handling Error Cases**

16867 The Query Next Image Response command SHALL have the disable default response bit set. Hence, if the command  
16868 is received successfully, no default response command SHALL be generated. However, the default response SHALL  
16869 be generated to indicate the error cases below.

#### 16870 **11.13.5.5.1 Malformed Command**

16871 Upon reception a badly formatted Query Next Image Response command, for example, the command is missing one  
16872 of the payload field, other payload fields are included when the status field is not SUCCESS, the image type value  
16873 included in the command does not match that of the device or the manufacturer code included in the command does  
16874 not match that of the device; the client SHOULD ignore the message and SHALL send default response command  
16875 with MALFORMED\_COMMAND status to the server.

---

<sup>134</sup> CCB 2464 reject same image

<sup>135</sup> CCB 2464 reject same image

<sup>136</sup> CCB 2464 reject same image

<sup>137</sup> CCB 2342 allow a client to reject a downgraded image

16876 **11.13.6 Image Block Request Command**

16877 **11.13.6.1 Payload Format**

16878 **Figure 11-13. Format of Image Block Request Command Payload**

Octets	1	2	2	4	4	1	0/8	0/2
Data Type	map8 <sup>138</sup>	uint16	uint16	uint32	uint32	uint8	EUI64	uint16
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Request node address	MinimumBlockPeriod

16879 **11.13.6.2 Payload Field Definitions**

16880 **11.13.6.2.1 Image Block Request Command Field Control**

16881 Field control value is used to indicate additional optional fields that MAY be included in the payload of Image Block  
 16882 Request command. Currently, the device is only required to support field control value of 0x00; support for other field  
 16883 control value is optional. A device SHALL process commands issued with unimplemented/unrecognized field control  
 16884 bits set. Devices SHALL correctly process messages containing fields indicated by unrecognized/unimplemented field  
 16885 control bits.

16886 Field control value 0x00 (bit 0 not set) indicates that the client is requesting a generic OTA upgrade file; hence, there  
 16887 is no need to include additional fields. The value of Image Type included in this case SHALL be manufacturer specific.

16888 Field control value of 0x01 (bit 0 set) means that the client’s IEEE address is included in the payload. This indicates  
 16889 that the client is requesting a device specific file such as security credential, log or configuration; hence, the need to  
 16890 include the device’s IEEE address in the image request command. The value of Image type included in this case  
 16891 SHALL be one of the reserved values that are assigned to each specific file type.

16892 **Table 11-20. Image Block Request Field Control Bitmask**

Bits	Name
0	Request node’s IEEE address Present
1	MinimumBlockPeriod present

16893 **11.13.6.2.2 Manufacturer Code**

16894 The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed  
 16895 description.

<sup>138</sup> CCB 2221 Field Control is a bitmap, not an unsigned integer

16896 **11.13.6.2.3 Image Type**

16897 The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed  
16898 description.

16899 **11.13.6.2.4 File Version**

16900 The file version included in the payload represents the OTA upgrade image file version that is being requested. See  
16901 section 11.4.2.7 for more detailed description.

16902 **11.13.6.2.5 File Offset**

16903 The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location  
16904 in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA  
16905 upgrade image file) data (in bytes) that the client has received so far.

16906 See section 11.10.2 for more description.

16907 **11.13.6.2.6 Maximum Data Size**

16908 The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server SHALL  
16909 respect the value and not send the data that is larger than the maximum data size. The server MAY send the data that  
16910 is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client  
16911 is multiple hops away. By having the client send both file offset and maximum data size in every command, it  
16912 eliminates the burden on the server for having to remember the information for each client.

16913 **11.13.6.2.7 Request Node Address (optional)**

16914 This is the IEEE address of the client device sending the Image Block Request command.

16915 **11.13.6.2.8 MinimumBlockPeriod (optional)**

16916 This is the current value of the *MinimumBlockPeriod* attribute of the device that is making the request as set by the  
16917 server. If the device supports the attribute, then it SHALL include this field in the request. The value is in  
16918 milliseconds<sup>139</sup>.

16919 This attribute does not necessarily reflect the actual delay applied by the client between Image Block Requests, only  
16920 the value set by the server on the client.

16921 **11.13.6.3 When Generated**

16922 The client device requests the image data at its leisure by sending Image Block Request command to the upgrade  
16923 server. The client knows the total number of request commands it needs to send from the image size value received in  
16924 Query Next Image Response command.

16925 The client repeats Image Block Requests until it has successfully obtained all data. Manufacturer code, image type  
16926 and file version are included in all further queries regarding that image. The information eliminates the need for the  
16927 server to remember which OTA Upgrade Image is being used for each download process.

16928 If the client supports the *MinimumBlockPeriod* attribute it SHALL include the value of the attribute as the  
16929 MinimumBlockPeriod field of the Image Block Request message. The client SHALL ensure that it delays at least  
16930 MinimumBlockPeriod after the previous Image Block Request was sent before sending the next Image Block Request  
16931 message. A client MAY delay its next Image Block Requests longer than its MinimumBlockPeriod attribute.

---

<sup>139</sup> CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

16932 **11.13.6.4 Effect on Receipt**

16933 The server uses the manufacturer code, image type, and file version to uniquely identify the OTA upgrade image  
 16934 request by the client. It uses the file offset to determine the location of the requested data within the OTA upgrade  
 16935 image. If the server supports rate-limited transfers it SHALL check the MinimumBlockPeriod<sup>140</sup> field and compare it  
 16936 to the desired rate for the client. If the server receives an Image Block Request with a field control mask of 0x02, (i.e.,  
 16937 MinimumBlockPeriod present) and the server does not support rate-limited transfers the server SHALL ignore the  
 16938 MinimumBlockPeriod value and process the command.

16939 **11.13.6.5 Handling Error Cases**

16940 In most cases, the server sends Image Block Response command in response to the client’s Image Block Request  
 16941 command. However, with the exception of a few error cases described below that the server SHALL send default  
 16942 response command as a response.

16943 **11.13.6.5.1 Malformed Command**

16944 Upon reception a badly formatted Image Block Request command, for example, the command is missing one of the  
 16945 payload field or the file offset value requested by the client is invalid, for example, the value is larger than the total  
 16946 image size; the server SHOULD ignore the message and it SHALL send default response command with  
 16947 MALFORMED\_COMMAND status to the client.

16948 **11.13.6.5.2 No Image Available**

16949 If either manufacturer code or image type or file version information in the request command is invalid or the OTA  
 16950 upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file,  
 16951 it SHALL send default response command with NO\_IMAGE\_AVAILABLE status to the client. After three attempts,  
 16952 if the client keeps getting the default response with the same status, it SHOULD go back to sending Query Next Image  
 16953 Request periodically or waiting for next Image Notify command.

16954 **11.13.6.5.3 Command Not Supported**

16955 If the client sends image request command with field control value of 0x01 that indicates device specific file request  
 16956 and if the server does not support such request, it SHALL send default response with  
 16957 UNSUP\_CLUSTER\_COMMAND status. Upon reception of such response, the client SHOULD terminate the attempt  
 16958 to request the device specific file and it MAY try to query different server.

16959 **11.13.7 Image Page Request Command**

16960 **11.13.7.1 Payload Format**

16961 **Figure 11-14. Image Page Request Command Payload**

Octets	1	2	2	4	4	1	2	2	0/8
Data Type	map8 <sup>141</sup>	uint16	uint16	uint32	uint32	uint8	uint16	uint16	EUI64

<sup>140</sup> CCB 2307 align attribute names with field names

<sup>141</sup> CCB 2219 Field Control is a bitmap, not an unsigned integer

Octets	1	2	2	4	4	1	2	2	0/8
Field Name	Field control	Manufacturer code	Image type	File version	File offset	Maximum data size	Page size	Response Spacing	Request node address

16962 **11.13.7.2 Payload Field Definitions**

16963 **11.13.7.2.1 Image Page Request Command Field Control**

16964 Field control value is used to indicate additional optional fields that MAY be included in the payload of Image Page  
16965 Request command. Currently, the device is only required to support field control value of 0x00; support for other field  
16966 control value is optional.

16967 Field control value 0x00 indicates that the client is requesting a generic OTA upgrade file; hence, there is no need to  
16968 include additional fields. The value of Image Type included in this case SHALL be manufacturer specific.

16969 Field control value of 0x01 means that the client’s IEEE address is included in the payload. This indicates that the  
16970 client is requesting a device specific file such as security credential, log or configuration; hence, the need to include  
16971 the device’s IEEE address in the image request command. The value of Image type included in this case SHALL be  
16972 one of the reserved values that are assigned to each specific file type.

16973 **Table 11-21. Image Page Request Field Control Bitmask**

Bits	Name
0	Request node’s IEEE address Present

16974 **11.13.7.2.2 Manufacturer Code**

16975 The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed  
16976 description.

16977 **11.13.7.2.3 Image Type**

16978 The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed  
16979 description.

16980 **11.13.7.2.4 File Version**

16981 The file version included in the payload represents the OTA upgrade image file version that is being requested. See  
16982 section 11.4.2.7 for more detailed description.

16983 **11.13.7.2.5 File Offset**

16984 The value indicates number of bytes of data offset from the beginning of the file. It essentially points to the location  
16985 in the OTA upgrade image file that the client is requesting the data from. The value reflects the amount of (OTA  
16986 upgrade image file) data (in bytes) that the client has received so far.

16987 See section 11.10.2 for more description.



16988 **11.13.7.2.6 Maximum Data Size**

16989 The value indicates the largest possible length of data (in bytes) that the client can receive at once. The server SHALL  
16990 respect the value and not send the data that is larger than the maximum data size. The server MAY send the data that  
16991 is smaller than the maximum data size value, for example, to account for source routing payload overhead if the client  
16992 is multiple hops away. By having the client send both file offset and maximum data size in every command, it  
16993 eliminates the burden on the server for having to remember the information for each client.

16994 **11.13.7.2.7 Page Size**

16995 The value indicates the number of bytes to be sent by the server before the client sends another Image Page Request  
16996 command. In general, page size value SHALL be larger than the maximum data size value.

16997 **11.13.7.2.8 Response Spacing**

16998 The value indicates how fast the server SHALL send the data (via Image Block Response command) to the client. The  
16999 value is determined by the client. The server SHALL wait at the minimum the (response) spacing value before sending  
17000 more data to the client. The value is in milliseconds.

17001 **11.13.7.2.9 Request Node Address (optional)**

17002 This is the IEEE address of the client device sending the Image Block Request command.

17003 **11.13.7.3 When Generated**

17004 The support for the command is optional. The client device MAY choose to request OTA upgrade data in one page  
17005 size at a time from upgrade server. Using Image Page Request reduces the numbers of requests sent from the client to  
17006 the upgrade server, compared to using Image Block Request command. In order to conserve battery life a device MAY  
17007 use the Image Page Request command. Using the Image Page Request command eliminates the need for the client  
17008 device to send Image Block Request command for every data block it needs; possibly saving the transmission of  
17009 hundreds or thousands of messages depending on the image size.

17010 The client keeps track of how much data it has received by keeping a cumulative count of each data size it has received  
17011 in each Image Block Response. Once the count has reach the value of the page size requested, it SHALL repeat Image  
17012 Page Requests until it has successfully obtained all pages. Note that the client MAY choose to switch between using  
17013 Image Block Request and Image Page Request during the upgrade process. For example, if the client does not receive  
17014 all data requested in one Image Page Request, the client MAY choose to request the missing block of data using Image  
17015 Block Request command, instead of requesting the whole page again.

17016 Since a single Image Page Request MAY result in multiple Image Block Response commands sent from the server,  
17017 the client, especially ZED client, SHOULD make its best effort to ensure that all responses are received. A ZED client  
17018 MAY select a small value for the response spacing and stay awake to receive all data blocks. Or it MAY choose a  
17019 larger value and sleeps between receiving each data block.

17020 Manufacturer code, image type and file version are included in all further queries regarding that image. The  
17021 information eliminates the need for the server to remember which OTA Upgrade Image is being used for each  
17022 download process.

17023 **11.13.7.4 Effect on Receipt**

17024 The server uses the file offset value to determine the location of the requested data within the OTA upgrade image.  
17025 The server MAY respond to a single Image Page Request command with possibly multiple Image Block Response  
17026 commands; depending on the value of page size. Each Image Block Response command sent as a result of Image Page  
17027 Request command SHALL have increasing ZCL sequence number. Note that the sequence number MAY not be  
17028 sequential (for example, if the server is also upgrading another client simultaneously); additionally ZCL sequence  
17029 numbers are only 8-bit and MAY wrap.

17030 In response to the Image Page Request, the server SHALL send Image Block Response commands with no APS retry  
17031 to disable APS acknowledgement. The intention is to minimize the number of packets sent by the client in order to  
17032 optimize the energy saving. APS acknowledgement is still used for Image Block Response sent in response to Image  
17033 Block Request command.

17034 Image Block Response message (in response to Image Page Request) only relies on network level retry. This MAY  
17035 not be as reliable over multiple hops communication, however, the benefit of using Image Page Request is to save  
17036 energy on the ZED client and using APS ack with the packet undermines that effort. ZED client needs to make the  
17037 decision which request it uses. Image Page Request MAY speed up the upgrade process; the client transmits fewer  
17038 packets, hence, less energy use but it MAY be less reliable. On the other hand, Image block request MAY slow down  
17039 the upgrade process; the client is required to transmit more packets but it is also more predictable and reliable; it also  
17040 allows the upgrade process to proceed at the client's pace.

### 17041 **11.13.7.5 Handling Error Cases**

17042 In most cases, the server sends Image Block Response command in response to the client's Image Page Request  
17043 command. However, with the exception of a few error cases described below that the server SHALL send default  
17044 response command as a response.

#### 17045 **11.13.7.5.1 Malformed Command**

17046 Upon reception a badly formatted Image Page Request command, for example, the command is missing one of the  
17047 payload fields or the file offset value requested by the client is invalid. The server SHOULD ignore the message and  
17048 it SHALL send default response command with MALFORMED\_COMMAND status to the client.

#### 17049 **11.13.7.5.2 No Image Available**

17050 If either manufacturer code or image type or file version information in the request command is invalid or the OTA  
17051 upgrade file for the client for some reason has disappeared which result in the server no longer able to retrieve the file,  
17052 it SHALL send default response command with NO\_IMAGE\_AVAILABLE status to the client. After three attempts,  
17053 if the client keeps getting the default response with the same status, it SHOULD go back to sending Query Next Image  
17054 Request periodically or waiting for next Image Notify command.

#### 17055 **11.13.7.5.3 Command Not Supported**

17056 If the client sends Image Page Request command with field control value of 0x00 to request OTA upgrade image and  
17057 the server does not support Image Page Request command, it SHALL send default response with  
17058 UNSUP\_CLUSTER\_COMMAND status. Upon reception of such response, the client SHALL switch to using Image  
17059 Block Request command instead to request OTA image data.

17060 If the client sends image request command with field control value of 0x01 that indicates device specific file request  
17061 and if the server does not support such request, it SHALL send default response with  
17062 UNSUP\_CLUSTER\_COMMAND status. Upon reception of such response, the client SHOULD terminate the attempt  
17063 to request the device specific file and it MAY try to query different server.

## 17064 **11.13.8 Image Block Response Command**

### 17065 **11.13.8.1 Payload Format**

17066 **Figure 11-15. Image Block Response Command Payload with SUCCESS status**

Octets	1	2	2	4	4	1	Variable
--------	---	---	---	---	---	---	----------

<b>Data Type</b>	enum8 <sub>142</sub>	uint16	uint16	uint32	uint32	uint8	Opaque
<b>Field Name</b>	Success status	Manufacturer code	Image type	File version	File offset	Data size	Image data

17067

17068

17069

17070

**Figure 11-16. Image Block Response Command Payload with WAIT\_FOR\_DATA status**

<b>Octets</b>	<b>1</b>	<b>4</b>	<b>4</b>	<b>2</b>
<b>Data Type</b>	enum8 <sup>143</sup>	uint32	uint32	uint16
<b>Field Name</b>	Wait for data Status	Current time	Request time	MinimumBlockPeriod

17071

17072

**Figure 11-17. Image Block Response Command Payload with ABORT status**

<b>Octets</b>	<b>1</b>
<b>Data Type</b>	enum8 <sup>144</sup>
<b>Field Name</b>	Abort Status

17073 **11.13.8.2 Payload Field Definitions**

17074 **11.13.8.2.1 Image Block Response Status**

17075 The status in the Image Block Response command MAY be SUCCESS, ABORT or WAIT\_FOR\_DATA. If the status  
 17076 is ABORT then only the status field SHALL be included in the message, all other fields SHALL be omitted.

17077 See section 11.13.2 for a complete list and description of OTA Cluster status codes.

17078 **11.13.8.2.2 Manufacturer Code**

17079 The value SHALL be the same as the one included in Image Block/Page Request command. See Chapter 2 for detailed  
 17080 description.

<sup>142</sup> CCB 2223 Status is an enumeration, not an unsigned integer

<sup>143</sup> CCB 2223 Status is an enumeration, not an unsigned integer

<sup>144</sup> CCB 2227 Status is an enumeration, not an unsigned integer

17081 **11.13.8.2.3 Image Type**

17082 The value SHALL be the same as the one included in Image Block/Page Request command. See section 11.4.2.6 for  
17083 detailed description.

17084 **11.13.8.2.4 File Version**

17085 The file version indicates the image version that the client is required to install. The version value MAY be lower than  
17086 the current image version on the client if the server decides to perform a downgrade. The version value MAY not be  
17087 the same as the client's current version. Reinstallation of the same software version is not supported. However, in  
17088 general, the version value SHOULD be higher than the current image version on the client to indicate an upgrade. See  
17089 section 11.4.2.7 for more description.

17090 **11.13.8.2.5 File Offset**

17091 The value represents the location of the data requested by the client. For most cases, the file offset value included in  
17092 the (Image Block) response SHOULD be the same as the value requested by the client. For (unsolicited) Image Block  
17093 responses generated as a result of Image Page Request, the file offset value SHALL be incremented to indicate the  
17094 next data location.

17095 **11.13.8.2.6 Data Size**

17096 The value indicates the length of the image data (in bytes) that is being included in the command. The value MAY be  
17097 equal or smaller than the maximum data size value requested by the client. See section 11.11.2 for more description.

17098 **11.13.8.2.7 Image Data**

17099 The actual OTA upgrade image data with the length equals to data size value. See section 11.11.3 for more description.

17100 **11.13.8.2.8 Current Time and Request Time**

17101 If status is WAIT\_FOR\_DATA, the payload then includes the server's current time and the request time that the client  
17102 SHALL retry the request command. The client SHALL wait at least the request time value before trying again. In case  
17103 of sleepy device, it MAY choose to wait longer than the specified time in order to not disrupt its sleeping cycle. If the  
17104 current time value is zero that means the server does not support UTC time and the client SHALL treat the request  
17105 time value as offset time. If neither time value is zero, and the client supports UTC time, it SHALL treat the request  
17106 time value as UTC time. If the client does not support UTC time, it SHALL calculate the offset time from the  
17107 difference between the two time values. The offset indicates the minimum amount of time to wait in seconds. The  
17108 UTC time indicates the actual time moment that needs to pass before the client SHOULD try again. See section 11.15.4  
17109 for more description.

17110 **11.13.8.2.9 MinimumBlockPeriod**

17111 This value is only included if the status is WAIT\_FOR\_DATA and the server supports rate limiting. This is the  
17112 minimum delay that the server wants the client to wait between subsequent block requests. The client SHALL update  
17113 its *MinimumBlockPeriod* attribute to this value. The MinimumBlockPeriod field value SHALL be observed in all  
17114 future Image Block Request messages for the duration of the firmware image download, or until updated by the server.

17115 If the server does not support rate limiting or does not wish to slow the client's download, the field SHALL be set to  
17116 0.

17117 The client SHALL check the existence of this field by looking at the length of the message. If the field does not exist,  
17118 then the field SHALL have the value of zero.<sup>145</sup>

---

<sup>145</sup> CCB 2296 this field was added as mandatory and older servers may not include it

17119 See 11.10.10 for more description of the valid ranges and use of this attribute.

17120 See section 11.15.3 for more description on how the rate limiting feature works.

### 17121 **11.13.8.3 When Generated**

17122 Upon receipt of an Image Block Request command the server SHALL generate an Image Block Response. If the  
17123 server is able to retrieve the data for the client and does not wish to change the image download rate, it will respond  
17124 with a status of SUCCESS and it will include all the fields in the payload. The use of file offset allows the server to  
17125 send packets with variable data size during the upgrade process. This allows the server to support a case when the  
17126 network topology of a client MAY change during the upgrade process, for example, mobile client MAY move around  
17127 during the upgrade process. If the client has moved a few hops away, the data size SHALL be smaller. Moreover,  
17128 using file offset eliminates the need for data padding since each Image Block Response command MAY contain  
17129 different data size. A simple server implementation MAY choose to only support largest possible data size for the  
17130 worst-case scenario in order to avoid supporting sending packets with variable data size.

17131 The server SHALL respect the maximum data size value requested by the client and SHALL not send the data with  
17132 length greater than that value. The server MAY send the data with length smaller than the value depending on the  
17133 network topology of the client. For example, the client MAY be able to receive 100 bytes of data at once so it sends  
17134 the request with 100 as maximum data size. But after considering all the security headers (perhaps from both APS and  
17135 network levels) and source routing overhead (for example, the client is five hops away), the largest possible data size  
17136 that the server can send to the client SHALL be smaller than 100 bytes.

17137 If the server simply wants to cancel the download process, it SHALL respond with ABORT status. An example is  
17138 while upgrading the client the server MAY receive newer image for that client. It MAY then choose to abort the  
17139 current process so that the client MAY reinitiate a new upgrade process for the newer image.

17140 If the server does not have the image block available for the client yet or it wants to slow down (pause or rate-limit)  
17141 the download process, it SHALL send the response back with status WAIT\_FOR\_DATA and with RequestTime value  
17142 that the client SHALL wait before resending the request. This is a one-time (temporary) delay of the download for the  
17143 client.

17144 If the Image Block Request message contains the MinimumBlockPeriod field and the server wishes to slow the client's  
17145 rate of sending Image Block requests, then the server SHALL send an Image Block Response with status  
17146 WAIT\_FOR\_DATA. In this case the RequestTime and CurrentTime in the message SHALL be set so that their delta  
17147 is zero, and the MinimumBlockPeriod field SHALL be set to the minimum delay that server wants the client to add  
17148 between all subsequent Image Block Requests.

### 17149 **11.13.8.4 Effect on Receipt**

17150 When the client receives the Image Block Response it SHALL examine the status field. If the value is SUCCESS, it  
17151 SHALL write the image data to its additional memory space. The client then SHALL continue to send Image Block  
17152 Request commands with incrementing block numbers to request the remaining blocks of the OTA upgrade image. If  
17153 the client has received the final block of the image, it SHALL generate an Upgrade End request command. In case of  
17154 the client using Image Page Request, after receiving an Image Block Response, the client SHALL wait for response  
17155 spacing time before expecting another Image Block Response from the server. A ZED client MAY go to sleep in  
17156 between receiving Image Block Responses in order to save the energy.

17157 If the client receives a response with ABORT status, it SHALL abort the upgrade process. It MAY retry the entire  
17158 upgrade operation at a later point in time.

17159 Upon receipt of WAIT\_FOR\_DATA status, the client SHALL wait at a minimum for the specified RequestTime and  
 17160 try to retrieve the image data again by resending Image Block Request or Image Page Request command with the  
 17161 same file offset value. If the CurrentTime and RequestTime are the same value and the client supports the  
 17162 *MinimumBlockPeriod* attribute, then it SHALL examine if the message contains the MinimumBlockPeriod<sup>146</sup> field in  
 17163 the Image Block Response. If the field is present and has a value is different than its current attribute value, it SHALL  
 17164 update its local attribute. Prior to sending its next Image Block Request message it SHALL add a minimum delay  
 17165 equal to the new value of its *MinimumBlockPeriod*<sup>147</sup> attribute.

17166 If the delta between the CurrentTime and RequestTime is zero and the MinimumBlockPeriod field is not present or is  
 17167 zero, the client MAY immediately send an Image Block Request command.

### 17168 11.13.8.5 Handling Error Cases

17169 If Image Block Response command is received successfully by the client, no default response will be generated if the  
 17170 disable default response bit is set in the ZCL header. However, a few error cases described below MAY cause the  
 17171 client to send default response to the server with an error code.

#### 17172 11.13.8.5.1 Malformed Command

17173 Upon reception a badly formatted Image Block Response command, for example, the command is missing one of the  
 17174 payload field, the payload fields do not correspond to the status field, the request time value returned by the server is  
 17175 invalid, for example, the value is less than the client’s current time or the value is less than the server’s own current  
 17176 time, the data size value returned by the server is invalid, for example, the value is greater than the maximum data  
 17177 size specified by the client, or the value does not match the number of bytes of data actually included in the payload,  
 17178 or the value, when combined with file offset, is greater than the total image size or the file offset value returned by the  
 17179 server is invalid. The client SHOULD ignore the command and SHALL send default response command with  
 17180 MALFORMED\_COMMAND status to the server.

## 17181 11.13.9 Upgrade End Request Command

### 17182 11.13.9.1 Payload Format

17183 **Figure 11-18. Format of Upgrade End Request Command Payload**

Octets	1	2	2	4
Data Type	enum8 <sup>148</sup>	uint16	uint16	uint32
Field Name	Status	Manufacturer code	Image type	File version

### 17184 11.13.9.2 Payload Field Definitions

#### 17185 11.13.9.2.1 Upgrade End Request Command Status

17186 The status value of the Upgrade End Request command SHALL be SUCCESS, INVALID\_IMAGE,  
 17187 REQUIRE\_MORE\_IMAGE, or ABORT. See section 11.13.2 for more description.

<sup>146</sup> CCB 2307 align attribute names with field names

<sup>147</sup> CCB 2307 align attribute names with field names

<sup>148</sup> CCB 2224 Status is an enumeration, not an unsigned integer

17188 **11.13.9.2.2 Manufacturer Code**

17189 The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed  
17190 description.

17191 **11.13.9.2.3 Image Type**

17192 The value SHALL be between 0x0000 - 0xffbf (manufacturer specific value range). See section 11.4.2.6 for detailed  
17193 description.

17194 **11.13.9.2.4 File Version**

17195 The file version included in the payload represents the newly downloaded OTA upgrade image file version. See section  
17196 11.4.2.7 for more detailed description.

17197 **11.13.9.3 When Generated**

17198 Upon reception all the image data, the client SHOULD verify the image to ensure its integrity and validity. If the  
17199 device requires signed images it SHALL examine the image and verify the signature as described in section 11.7.1.  
17200 Clients MAY perform additional manufacturer specific integrity checks to validate the image, for example, CRC check  
17201 on the actual file data.

17202 If the image fails any integrity checks, the client SHALL send an Upgrade End Request command to the upgrade  
17203 server with a status of INVALID\_IMAGE. In this case, the client MAY reinitiate the upgrade process in order to  
17204 obtain a valid OTA upgrade image. The client SHALL not upgrade to the bad image and SHALL discard the  
17205 downloaded image data.

17206 If the image passes all integrity checks and the client does not require additional OTA upgrade image file, it SHALL  
17207 send back an Upgrade End Request with a status of SUCCESS. However, if the client requires multiple OTA upgrade  
17208 image files before performing an upgrade, it SHALL send an Upgrade End Request command with status  
17209 REQUIRE\_MORE\_IMAGE. This SHALL indicate to the server that it cannot yet upgrade the image it received.

17210 If the client decides to cancel the download process for any other reasons, it has the option of sending Upgrade End  
17211 Request with status of ABORT at any time during the download process. The client SHALL then try to reinitiate the  
17212 download process again at a later time.

17213 When a client finishes downloading a device specific file, it SHALL send Upgrade End Request command with status  
17214 of SUCCESS to the server to indicate the end of the upgrade process.

17215 **11.13.9.4 Effect on Receipt**

17216 For manufacturer specific image type file download, upon receipt of a SUCCESS Upgrade End Request command the  
17217 upgrade server SHALL reply with the Upgrade End Response indicating when the client SHALL upgrade to the newly  
17218 retrieved image. For other status value received such as INVALID\_IMAGE, REQUIRE\_MORE\_IMAGE, or  
17219 ABORT, the upgrade server SHALL not send Upgrade End Response command but it SHALL send default response  
17220 command with status of success and it SHALL wait for the client to reinitiate the upgrade process.

17221 The server MAY utilize the Upgrade End Request command as a means to know when devices are done downloading  
17222 a particular image. This helps the server manage the images and remove those that are no longer needed. However,  
17223 the upgrade server SHOULD not rely on receiving the command and MAY impose upper limits on how long it will  
17224 store a particular OTA upgrade image. The specific implementation of this is outside the scope of this document.

17225 **11.13.9.5 Handling Error Cases**

17226 Upgrade End Request command does not have disable default response bit set. Hence, in a case where the Upgrade  
17227 End Request command has been received and the server does not send Upgrade End Response command in response,  
17228 a default response command SHALL be sent with SUCCESS status. If the Upgrade End Request command has not  
17229 been received, default response command with error status SHALL be sent as described below.

17230 **11.13.9.5.1 Malformed Command**

17231 Upon reception a badly formatted Upgrade End Request command, for example, the command is missing one of the  
17232 payload fields. The server SHALL send default response command with MALFORMED\_COMMAND status to the  
17233 client.

17234 **11.13.9.6 Upgrade End Response Command**

17235 **11.13.9.6.1 Payload Format**

17236 **Figure 11-19. Format of Upgrade End Response Command Payload**

<b>Octets</b>	2	2	4	4	4
<b>Data Type</b>	uint16	uint16	uint32	UTC <sup>149</sup>	UTC <sup>150</sup>
<b>Field Name</b>	Manufacturer code	Image type	File version	Current time	Upgrade time

17237 **11.13.9.6.2 Payload Field Definitions**

17238 The ability to send the command with wild card values for manufacturer code, image type and file version is useful in  
17239 this case because it eliminates the need for the server having to send the command multiple times for each  
17240 manufacturer as well as having to keep track of all devices' manufacturers in the network.

17241 **11.13.9.6.3 Manufacturer Code**

17242 Manufacturer code MAY be sent using wildcard value of 0xffff in order to apply the command to all devices disregard  
17243 of their manufacturers. See Chapter 2 for detailed description.

17244 **11.13.9.6.4 Image Type**

17245 Image type MAY be sent using wildcard value of 0xffff in order to apply the command to all devices disregard of  
17246 their manufacturers. See section 11.4.2.6 for detailed description.

<sup>149</sup> CCB 2226 UTC time, not unsigned 32-bit integer

<sup>150</sup> CCB 2225 UTC time, not unsigned 32-bit integer



17247 **11.13.9.6.5 File Version**

17248 The file version included in the payload represents the newly downloaded OTA upgrade image file version. The value  
17249 SHALL match that included in the request. Alternatively, file version MAY be sent using wildcard value of 0xffffffff  
17250 in order to apply the command to all devices disregard of their manufacturers. See section 11.4.2.7 for more detailed  
17251 description.

17252 **Current Time and Upgrade Time**

17253 Current time and Upgrade time values are used by the client device to determine when to upgrade its running firmware  
17254 image(s) with the newly downloaded one(s). See section 11.15.4 for more description.

17255 **11.13.9.7 When Generated**

17256 When an upgrade server receives an Upgrade End Request command with a status of INVALID\_IMAGE,  
17257 REQUIRE\_MORE\_IMAGE, or ABORT, no additional processing SHALL be done in its part. If the upgrade server  
17258 receives an Upgrade End Request command with a status of SUCCESS, it SHALL generate an Upgrade End Response  
17259 with the manufacturer code and image type received in the Upgrade End Request along with the times indicating when  
17260 the device SHOULD upgrade to the new image.

17261 The server MAY send an unsolicited Upgrade End Response command to the client. This MAY be used for example  
17262 if the server wants to synchronize the upgrade on multiple clients simultaneously. For client devices, the upgrade  
17263 server MAY unicast or broadcast Upgrade End Response command indicating a single client device or multiple client  
17264 devices SHALL switch to using their new images. The command MAY not be reliably received by sleepy devices if  
17265 it is sent unsolicited.

17266 For device specific file download, the client SHOULD not always expect the server to respond back with Upgrade  
17267 End Response command. For example, in a case of a client has just finished retrieving a log file from the server, the  
17268 server MAY not need to send Upgrade End Response command. However, if the client has just retrieved a security  
17269 credential or a configuration file, the server MAY send Upgrade End Response command to notify the client of when  
17270 to apply the file. The decision of whether Upgrade End Response command SHOULD be sent for device specific file  
17271 download is manufacturer specific.

17272 **11.13.9.8 Effect on Receipt**

17273 The client SHALL examine the manufacturer code, image type and file version to verify that they match its own. If  
17274 the received values do not match its own values or they are not wild card values, then it SHALL discard the command  
17275 and no further processing SHALL continue. If all values match, the client SHALL examine the time values to  
17276 determine the upgrade time. For more information on determining the time, please refer to section 11.15.4.

17277 **11.13.9.9 Handling Error Cases**

17278 If Upgrade End Response command is received successfully by the client or if it is sent as broadcast or multicast, no  
17279 default response will be generated. However, a few error cases described below MAY cause the client to send default  
17280 response to the server.

17281 **11.13.9.9.1 Malformed Command**

17282 Upon reception a badly formatted Upgrade End Response command, for example, the command is missing one of the  
17283 payload field or the request time value returned by the server is invalid, for example, the value is less than the client's  
17284 current time or the value is less than the server's own current time. The client SHOULD ignore the command and  
17285 SHALL send default response command with MALFORMED\_COMMAND status to the server.

17286 **11.13.10 Query Device Specific File Request Command**17287 **11.13.10.1 Payload Format**17288 **Figure 11-20. Format of Query Device Specific File Request Command Payload**

<b>Octets</b>	8	2	2	4	2
<b>Data Type</b>	EUI64	uint16	uint16	uint32	uint16
<b>Field Name</b>	Request node address	Manufacturer code	Image type	File version	(Current) ZigBee stack version

17289 **11.13.10.2 Payload Field Definitions**17290 **11.13.10.2.1 Request Node Address**

17291 This is the IEEE address of the client device sending the request command. This indicates that the client is requesting  
17292 a device specific file such as security credential, log or configuration; hence, the need to include the device's IEEE  
17293 address in the image request command.

17294 **11.13.10.2.2 Manufacturer Code**

17295 The value SHALL be that of the client device assigned to each manufacturer by ZigBee. See Chapter 2 for detailed  
17296 description.

17297 **11.13.10.2.3 Image Type**

17298 The value of image type included in this case SHALL be one of the reserved values that are assigned to each specific  
17299 file type. The value SHOULD be between 0xffc0 – 0xfffe, however, only 0xffc0 – 0xffc2 is being used currently. See  
17300 section 11.4.2.6 for detailed description.

17301 **11.13.10.2.4 File Version**

17302 The value indicates the version of the device specific file being requested. See section 11.4.2.7 for more detailed  
17303 description.

17304 **11.13.10.2.5 (current) ZigBee Stack Version**

17305 The value MAY represent the current running ZigBee stack version on the device or the ZigBee stack version of the  
17306 OTA upgrade image being stored in additional memory space. The decision of which value to include depends on  
17307 which device specific file being requested. For example, if the client is requesting a new security credential file in  
17308 order to be able to run the newly downloaded image (ex. SE 2.0), then it SHOULD include the ZigBee stack version  
17309 value of the new image.

17310 **11.13.10.3 When Generated**

17311 Client devices SHALL send a Query Device Specific File Request command to the server to request for a file that is  
 17312 specific and unique to it. Such file could contain non-firmware data such as security credential (needed for upgrading  
 17313 from Smart Energy 1.1 to Smart Energy 2.0), configuration or log. When the device decides to send the Query Device  
 17314 Specific File Request command is manufacturer specific. However, one example is during upgrading from SE 1.1 to  
 17315 2.0 where the client MAY have already obtained new SE 2.0 image and now needs new SE 2.0 security credential  
 17316 data.

17317 The fields included in the payload helps the upgrade server in obtaining or creating the right file for the client.

17318 **11.13.10.4 Effect on Receipt**

17319 The server takes the client’s information in the command and either obtain the file via the backend system or create  
 17320 the file itself. Details of how the file is being obtained or created is manufacturer specific and outside the scope of this  
 17321 document. The device specific file SHALL follow OTA upgrade file format (section 11.3) and SHALL have Device  
 17322 Specific File bit set in OTA header field control. Moreover, the value of the Upgrade File Destination field in the OTA  
 17323 header SHALL match the Request node address value in the command’s field.

17324 **11.13.10.5 Handling Error Cases**

17325 In most cases all error cases resulted from receiving Query Device Specific File Request command are handled by the  
 17326 corresponding Query Device Specific File Response command with the exception of a few error cases described below  
 17327 that are handled by default response command.

17328 **11.13.10.5.1 Malformed Command**

17329 Upon reception a badly formatted Query Device Specific File Request command, for example, the command is  
 17330 missing one of the payload fields; the server SHALL send default response command with  
 17331 MALFORMED\_COMMAND status to the client and it SHALL not process the command further.

17332 **11.13.10.5.2 Command Not Supported**

17333 Certain server MAY not support transferring of device specific file and the implement of Query Device Specific File  
 17334 Request command; in this case the server SHALL send default response with UNSUP\_CLUSTER\_COMMAND  
 17335 status.

17336 **11.13.11 Query Device Specific File Response Command**

17337 **11.13.11.1 Payload Format**

17338 **Figure 11-21. Format of Query Device Specific File Response Command Payload**

Octets	1	0/2	0/2	0/4	0/4
Data Type	enum <sup>151</sup>	uint16	uint16	uint32	uint32
Field Name	Status	Manufacturer code	Image type	File version	Image size

<sup>151</sup> CCB 2227 Status is an enumeration, not an unsigned integer

## 17339 **11.13.11.2 Payload Field Definitions**

### 17340 **11.13.11.2.1 Query Device Specific File Response Status**

17341 Only if the status is SUCCESS that other fields are included. For other (error) status values, only status field SHALL  
17342 be present.

### 17343 **11.13.11.2.2 Manufacturer Code**

17344 The value SHALL be the one received by the server in the Query Device Specific File Request command. See Chapter  
17345 2 for detailed description.

### 17346 **11.13.11.2.3 Image Type**

17347 The value SHALL be the one received by the server in the Query Device Specific File Request command. See section  
17348 11.4.2.6 for detailed description.

### 17349 **11.13.11.2.4 File Version**

17350 The file version indicates the image version that the client is required to download. The value SHALL be the same as  
17351 the one included in the request. See section 11.4.2.7 for more description.

### 17352 **11.13.11.2.5 Image Size**

17353 The value represents the total size of the image (in bytes) including all sub-elements. See section 11.4.2.10 for more  
17354 description.

## 17355 **11.13.11.3 When Generated**

17356 The server sends Query Device Specific File Response after receiving Query Device Specific File Request from a  
17357 client. The server SHALL determine whether it first supports the Query Device Specific File Request command. Then  
17358 it SHALL determine whether it has the specific file being requested by the client using all the information included in  
17359 the request. The upgrade server sends a Query Device Specific File Response with one of the following status:  
17360 SUCCESS, NO\_IMAGE\_AVAILABLE or NOT\_AUTHORIZED.

17361 A status of NO\_IMAGE\_AVAILABLE indicates that the server currently does not have the device specific file  
17362 available for the client. A status of NOT\_AUTHORIZED indicates the server is not authorized to send the file to the  
17363 client.

## 17364 **11.13.11.4 Effect on Receipt**

17365 A status of SUCCESS in the Query Device Specific File response indicates to the client that the server has a specific  
17366 file for it. The client SHALL begin requesting file data using the Image Block Request or Image Page Request  
17367 command with a field control value set to 0x01 and include its IEEE address. A ZED client MAY choose to change  
17368 its wake cycle to retrieve the file more quickly.

17369 If the client receives the response with status of NOT\_AUTHORIZED, it MAY perform discovery again to find  
17370 another upgrade server. The client MAY implement an intelligence to avoid querying the same unauthorized server.

## 17371 **11.13.11.5 Handling Error Cases**

17372 Query Device Specific File Response command SHALL have disable default response bit set. Hence, if the command  
17373 is received successfully, no default response command SHALL be generated. However, the default response SHALL  
17374 be generated to indicate the error cases below.

17375 **11.13.11.5.1 Malformed Command**

17376 Upon reception a badly formatted Query Device Specific File Response command, for example, the command is  
17377 missing one of the payload field, other payload fields are included when the status field is not SUCCESS, the  
17378 manufacturer code included in the command does not match that of the device or the image type value included in the  
17379 command does not match that of the device; the client SHOULD ignore the message and SHALL send default response  
17380 command with MALFORMED\_COMMAND status to the server.

17381 **11.14 Multiple Files Required for a Bootload**

---

17382 ZigBee devices MAY require multiple bootload files in order to be upgraded correctly. These files often correspond  
17383 to multiple embedded chips contained within the physical device that have separate firmware images to run them.

17384 A device has a number of options for managing these files depending on its own internal configuration or  
17385 dependencies. This section describes the three main options:

17386 **11.14.1 Single OTA File with multiple sub-elements**

17387 One of the simplest mechanisms to support multiple firmware images is to bundle all the images into a single OTA  
17388 file. Within the OTA file each firmware image could be noted with a different sub-element tag indicating the module  
17389 it is designated for. The advantage of this system is that it allows for a single OTA client to request a single OTA file  
17390 from the server that contains all the upgrade data it needs. Management of the multiple firmware images is handled  
17391 internally by the device.

17392 Typically, a manufacturer would put all of the firmware images used by the device into the image and upgrade all  
17393 modules at the same time. In that case the device manufacturer would need a download storage space (e.g. an external  
17394 EEPROM) big enough to hold an OTA image that contained all the firmware images for all the modules.

17395 The OTA client reports only the overall upgrade status regardless of how many internal modules are being  
17396 manipulated. The OTA client's attributes reflect only the single OTA *Image Type ID*, *CurrentFileVersion*,  
17397 *DownloadedVersion*, and *ImageUpgradeStatus* attributes.

17398 **11.14.2 Separate OTA Files Upgraded Independently**

17399 Another method that can be used is to have each upgradeable module within the physical device request bootload  
17400 images from the OTA server separately. In this case a module would report the same manufacturer ID but a different  
17401 image type ID. The modules would operate on separate endpoints to properly report the attributes about the current  
17402 state of that module's upgrade cycle (*ImageUpgradeStatus*) as well as the version number it is running  
17403 (*CurrentFileVersion*) and downloading (*DownloadedFileVersion*). As each module completed a download they would  
17404 separately request permission to finish the upgrade via the *Upgrade End Request command*.

17405 During the manufacturer specific part of the upgrade, it is possible that the OTA client endpoint undergoing the  
17406 upgrade, or even the entire ZigBee NWK layer, MAY not be accessible over-the-air. Once the upgrade is complete  
17407 the endpoint's client attributes reflecting the new version would be updated.

17408 Manufacturers are free to choose different versioning schemes for each image type used by the physical device and  
17409 decide when to release updates for each module. However, in general it is assumed that each module can be upgraded  
17410 independently of the others. Each OTA file would need to be given to the OTA server and managed separately.

17411 Though each module operates independently it is certainly possible that specific, shared resources MAY preclude  
17412 multiple simultaneous downloads or upgrades. For example, if the device has a single EEPROM that can store only  
17413 one download image at a time, then only one OTA client MAY be downloading or updating. Other OTA clients on  
17414 other endpoints corresponding to other modules would have to wait until the required resources are free for it to use.

### 17415 **11.14.3 Multiple OTA Files Dependent on Each Other**

17416 The last method a device might use to handle upgrading separate modules in the physical device is to use multiple  
17417 OTA files that have a dependency on each other. In this case the OTA client would sequentially download and apply  
17418 each OTA file before going to the next one.

17419 This method might be used in the case where a single OTA file containing all the OTA images is not possible because  
17420 the device does not contain a storage space big enough to hold all the module firmware images. Additionally, each  
17421 module cannot operate independently due to an internal device restriction.

17422 The details of the dependencies within the OTA files are specific to the manufacturer of the device. For example, if  
17423 the device required the OTA file for Image Type 7 before it received the OTA file for Image Type 3, the device must  
17424 manage this.

17425 After each OTA file has been downloaded and processed the OTA client SHALL send an Upgrade End Request  
17426 command with a status of REQUIRE\_MORE\_IMAGE. It SHALL then download and process the next file. In this  
17427 case the act of “processing” is manufacturer specific; it MAY or MAY NOT involve upgrading the internal  
17428 component. During each OTA file download the OTA client SHALL update its attributes to reflect the module that is  
17429 being upgraded. For example, the Image Type ID, CurrentFileVersion, DownloadedFileVersion SHALL be set to the  
17430 values of the internal module that the OTA client is processing an upgrade image for.

17431 Upon completion of the download for all modules the OTA client SHALL send an Upgrade End Request command  
17432 with a status of SUCCESS. The OTA server has the ability to delay or abort the final upgrade via the normal  
17433 mechanisms.

## 17434 **11.15 OTA Upgrade Cluster Management**

17435 This section provides ways for the upgrade server to monitor and manage the network-wide OTA upgrade process. It  
17436 is important to realize that the server cannot reliably query the upgrade status of the sleepy devices.

### 17437 **11.15.1 Query Upgrade Status**

17438 Server MAY send ZCL read attribute command for Image Upgrade Status attribute on the client devices. The attribute  
17439 indicates the progress of the client’s file download as well as its upgrade progress. The server MAY want to make  
17440 sure that all clients have completely downloaded their new images prior to issuing the Upgrade End Response  
17441 command.

17442 A client SHALL only download a single file at a time. It SHALL not download a second file while the first file  
17443 download is incomplete. This insures that the values in the client’s attributes can be correlated to a single download  
17444 instance.

### 17445 **11.15.2 Query Downloaded ZigBee Stack and File 17446 Versions**

17447 The server MAY send ZCL read attribute command to a client to determine its downloaded ZigBee stack version and  
17448 file version. The server SHOULD make sure that the client has downloaded the correct image prior to issuing the  
17449 Upgrade End Response command.

### 17450 **11.15.3 Rate Limiting**

17451 The OTA Upgrade Cluster server can rate limit how quickly clients download files by setting the  
17452 *MinimumBlockPeriod* attribute<sup>152</sup>. This feature is only available if the client supports the attribute, and the server  
17453 supports this optional feature. Client support can be determined by requesting the *MinimumBlockPeriod*<sup>153</sup> attribute  
17454 from the client, or if the Image Block Request message contains the *MinimumBlockPeriod*<sup>154</sup> field.

17455 The server has the ability to set the attribute while the client is downloading by responding to any Image Block Request  
17456 with an Image Block Response with a status of *WAIT\_FOR\_DATA*. The Image Block Response SHALL include the  
17457 Block Request field with the new delay desired by the server for all the client's subsequent requests. Upon receipt of  
17458 the Image Block Response the client will record the new value in its local *MinimumBlockPeriod* attribute and use it  
17459 for the rest of the download.

17460 The server can change the download delay of the client multiple times over the course of the download based on  
17461 whatever criteria it deems appropriate. For example, if the server detects only 1 client is downloading, it could allow  
17462 that client to download at full speed (*MinimumBlockPeriod* = 0), but if other clients simultaneously start downloads  
17463 it could limit all clients to 1 Image Block Request every 500 milliseconds<sup>155</sup>. Alternatively, it could give higher priority  
17464 to certain clients to download their upgrade image and let them download at full speed, while slowing down other  
17465 clients.

17466 The *MinimumBlockPeriod* attribute is a minimum delay. The client MAY request data slower than what the server  
17467 specifies (i.e. with a longer delay). Sleeping end devices MAY do this normally to conserve battery power.

17468 Below is a diagram showing how the rate limiting process generally works.

17469

---

<sup>152</sup> CCB 2307 align attribute names with field names

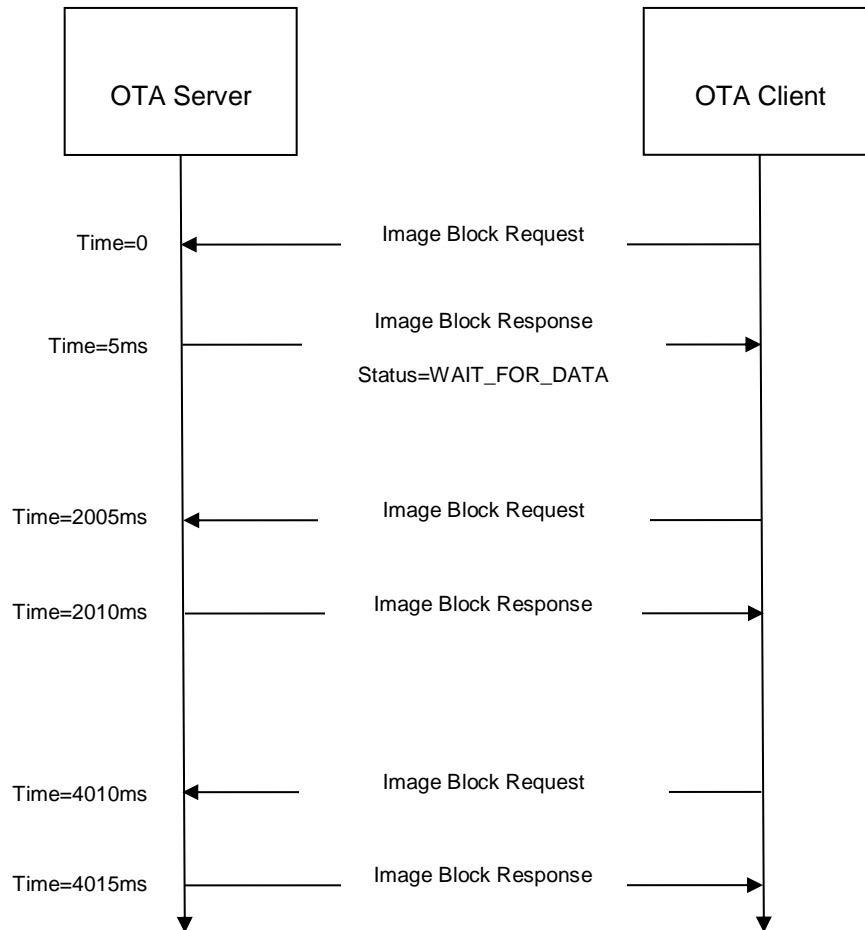
<sup>153</sup> CCB 2307 align attribute names with field names

<sup>154</sup> CCB 2307 align attribute names with field names

<sup>155</sup> CCB 2398 *MinimumBlockPeriod* is milliseconds (not seconds)

17470

**Figure 11-22. Rate Limiting Exchange**



17471

#### 11.15.4 Current Time, Request Time, and MinimumBlockPeriod

17472

17473

17474 When a server sends an Image Block Response with a status of WAIT\_FOR\_DATA, it can delay the client’s next  
17475 Image Block Request. This can be done persistently for all subsequent requests, or temporarily as a onetime delay.

17476 The onetime delay can be created by setting the Current Time and Request Time fields as described in section  
17477 11.13.8.2.8. This might occur if the server does not immediately have access to the block of the upgrade image  
17478 requested by the client, and the server must fetch the block from another location.

17479 The persistent delay can be enabled by setting the MinimumBlockPeriod as described in section 11.13.6.2.8 however  
17480 this only works if the client and server support this functionality.

17481



17482

## 11.16 OTA Upgrade Process

---

17483 Once a device has completely downloaded the image and returned a status of SUCCESS in the Upgrade End Request,  
17484 it SHALL obey the server's directive based on when it SHOULD upgrade. However, there are many failure scenarios  
17485 where this MAY not be possible. In such failure case, the device SHOULD attempt to contact the server and determine  
17486 what SHOULD be done, but if that has failed as well, then it MAY apply its update without an explicit command by  
17487 the server.

17488 After receiving an Upgrade End Response from the server the client will apply the upgrade according to time values  
17489 specified in the message. If the response directs the device to wait forever, it SHALL periodically query the server  
17490 about when it SHOULD apply the new upgrade. This SHALL happen at a period no more often than once every 60  
17491 minutes. If the server is unreachable after 3 retries, the device MAY apply the upgrade (see section 11.11.4 for further  
17492 details).

17493 The client does not need to persistently store the time indicating when to apply the upgrade. If the client feels that it  
17494 has lost connection to the upgrade server, it SHALL first try to rediscover the upgrade server perhaps by rejoining to  
17495 the network and performing network address discovery using the stored UpgradeServerID attribute. Once the server  
17496 is found, the client SHALL resend an Upgrade End Request command with a status of SUCCESS to the server,  
17497 including the relevant upgrade file information. The server SHALL send it a response again indicating when it  
17498 SHOULD upgrade. If the device is unable to communicate to the upgrade server or it cannot synchronize the time, it  
17499 MAY apply the upgrade anyway.

17500 When the time comes for the client to upgrade, the device SHOULD begin the manufacturer specific method to  
17501 upgrade its image. The upgrade MAY involve one or more hardware resets. Once the device has completed the  
17502 upgrade it SHOULD be able to reinitialize itself and start communicating on the network again. Previous network  
17503 information such as channel, power, short pan id, extended pan id SHOULD be preserved across the upgrade.

17504

## 11.17 Application Standard Specific Decisions

---

17505 Below are the decisions that each application standard needs to make in order to ensure successful OTA upgrade of  
17506 devices in the network.

- 17507
- The following are security considerations that SHOULD be taken into account when using this cluster.
    - Whether image signatures will be used to sign the OTA upgrade file. If a signature is used, what type of image signature will it be (example: ECDSA).
    - What encryption will be used during the transport of OTA data
  - Whether to use offset or UTC time in Image Block Response and Upgrade End Response commands. Refer to sections 11.11.4 and 11.13.9 for more details. If the application standard does not specify which type of (OTA upgrade) time to support, it is default to using the offset time since it does not require an implementation of ZCL time cluster. Once the application standard has decided which type of time to support, only that type of time SHALL be used consistently across the OTA upgrading. The standard SHALL avoid using both types of time simultaneously to avoid any confusion and inconsistency between the two time values.

17517 Other application standard wide decisions that SHOULD be answered are:

- 17518
- How often the OTA client SHALL discover the OTA server until it finds one that is authorized to do the upgrade.
  - How often the ZED client SHALL query the OTA server for new OTA upgrade image.
  - How often the ZED devices SHALL query for image data.
- 17519
- 17520
- 17521

### 17522 **11.17.1 SE Profile Standard: OTA Upgrade from SE 1.x to** 17523 **SE 2.0**

17524 The definition of SE Profile 2.0 is currently still being worked on by the ZigBee SE group. However, it is suggested  
17525 that in order to successfully upgrade a device from SE 1.x to SE 2.0, such process MAY involve transferring new  
17526 security data over-the-air from the server to the client device. OTA Upgrade cluster has provided a set of commands  
17527 that MAY be used to obtain such security data. The security data will be requested separately by the client using Query  
17528 Device Specific File Request command. The data is sent from the server to the client as an OTA upgrade file via  
17529 similar set of commands used to request firmware image. This OTA security file will be specific to a particular client  
17530 device.

17531 A client SHOULD request new security data necessary for SE Profile 2.0 via Query Device Specific File Request  
17532 command. After obtaining the security data file, the server will include the file information in the Query Device  
17533 Specific File Response command in response to the client's request. Upon reception the response, the client then  
17534 SHALL obtain the file via Image Block or Page Request command. Query Device Specific File Request and Response  
17535 commands are described in sections 11.13.10 and 11.13.11 respectively.

## 17536 **11.18 OTA Upgrade Recovery**

17537 Each manufacturer is encouraged to implement a recovery method that SHOULD be used to recover the node in a  
17538 case when the OTA upgrade fails. The recovery method is particularly important in a case where the device MAY not  
17539 be able to communicate to the server over-the-air. The actual recovery implementation is manufacturer specific;  
17540 however, some of the options are discussed in this section.

17541 One option for recovery method is the ability for the application bootloader to swap the images between its external  
17542 flash and its internal flash, rather than just overwriting the internal with the external. A sample use case is where the  
17543 upgraded device is functional enough to receive a message, but broken enough to not be able to initiate OTA upgrade  
17544 process again. A manufacturer specific command MAY be sent from the server to notify the device to revert back to  
17545 its previous image.

17546 In a case where the device is no longer able to communicate to the server over-the-air; the application bootloader  
17547 could revert to the previous image via a button press on power up.

# CHAPTER 12 TELECOMMUNICATION

17548  
 17549 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 17550 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 17551 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 17552 using [*Rn*] notation.

## 12.1 General Description

### 12.1.1 Introduction

17554  
 17555 The clusters specified in this chapter are for use typically in telecommunication applications but may be used in any  
 17556 application domain.

### 12.1.2 Cluster List

17557  
 17558 This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of  
 17559 clarification. The clusters specified in this chapter are listed in Table 12-1.

Table 12-1. Telecom Cluster List

ID	Cluster Name	Description
0x0900	Information	Commands and attributes for information delivery
0x0905	Chatting	Commands and attributes for sending chat messages
0x0904	Voice Over ZigBee	Commands and attributes for voice receiving and transmitting

## 12.2 Information

### 12.2.1 Scope and Purpose

17562  
 17563 This section specifies the Information cluster, which provides commands and attributes for information delivery  
 17564 service on ZigBee networks and also specifies three types of special nodes on which this cluster works. One of the  
 17565 nodes, Information Node (IN) is a node which provides information contents in both pull-based and push-based  
 17566 information delivery to a mobile terminal. The contents may have links to other contents and thus they may be  
 17567 organized in a structure. Mobile Terminal (MT) is the one of special nodes and is used by an end-user who looks into  
 17568 information from the information node. The other node is Access Point (AP) which updates contents stored in  
 17569 Information nodes and has a role of gateway connected to the operator network. It is also assumed to be a ZigBee  
 17570 coordinator which forms a network with Information nodes and Mobile Terminals. Access point may have a function  
 17571 of Information Node.

17572 This section should be used in conjunction with the Foundation Chapter, which gives an overview of the library and  
 17573 specifies the frame formats and general commands used therein.

17574 Information Delivery Service in this document is considered ‘Pull-based delivery’ and ‘Push-based delivery’. Both  
 17575 methods are provided by a single cluster, the information cluster.

17576 Figure 12-2 shows typical usage of the cluster. This cluster may use Partition cluster.

17577 **12.2.1.1 Data Structure of Contents Data**

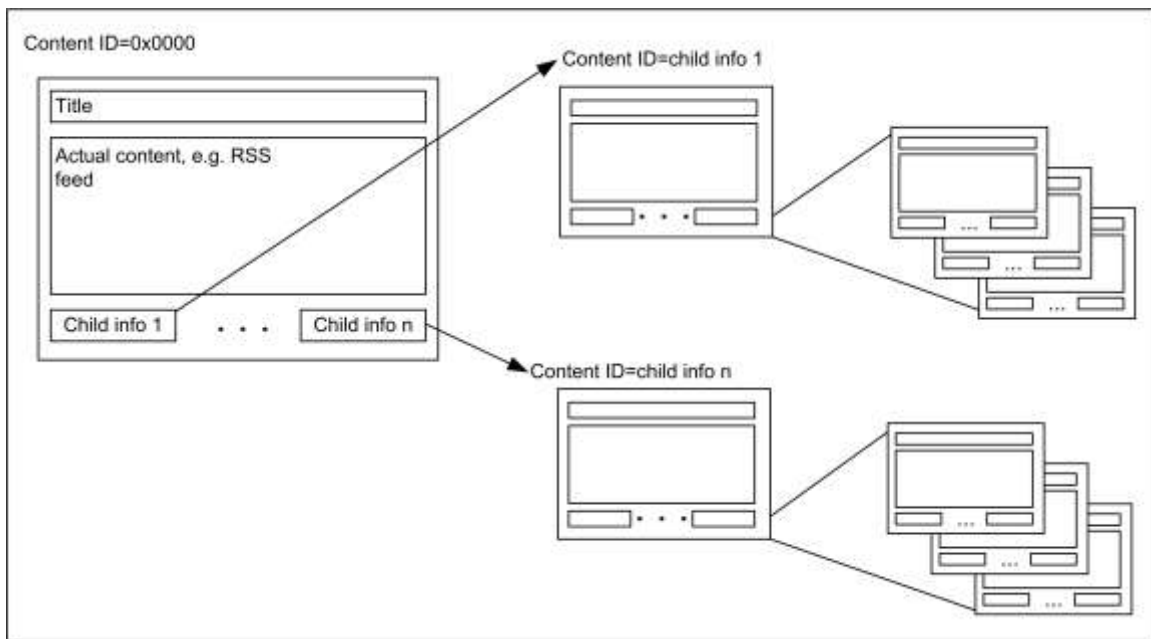
17578 Typical data structure of contents data is as illustrated in Figure 12-1. Each content data has its Content ID, which is  
17579 used when the client cluster requests content to the server cluster.

17580 A content data includes ‘the title strings’, ‘actual content’, ‘number of child contents’ and ‘children’s content IDs’.

17581 To let each content data have its children content data makes it enables to organize list-structure or tree-structure and  
17582 obtain a hierarchical content data structure, and a user who uses information delivery can request information along  
17583 to child information links.

17584 To obtain the first contents ID, there are methods, reading server attribute ‘Root ID’, using ID sent by out-of-band  
17585 like via GPRS network and using ID provided by another telecom application clusters (ex. Payment or gaming).

17586 **Figure 12-1. Typical Content Data Structure**



17587

17588 **12.2.2 Cluster List**

17589 A cluster specified in this document is listed in Table 12-2.

17590 Server cluster is expected to be implemented in the Information Node. Client cluster (including functions related to  
17591 contents provisioning) is expected to be implemented in the Mobile Terminal. A part of commands of client cluster,  
17592 update command and configuration commands are expected to be implemented in the Access Point. The Access Point  
17593 may have functionality of the Information node and it has server cluster in that case. Some user specific content is  
17594 provided with processing user-side information, defined as a preference. If a preference needs to be processed not in  
17595 the Information Node but in an Access Point or in a server beyond the Access Point as a gateway, information indicates  
17596 the Access Point’s ID so that the Mobile Terminal can switch to access it (as illustrated in Figure 12-3) or the  
17597 Information Node acts as proxy and access the Access Point with client function to forward preference, commands to  
17598 the Access Point and contents to the Mobile Terminal (as illustrated in

17599 Figure 12-2).

17600

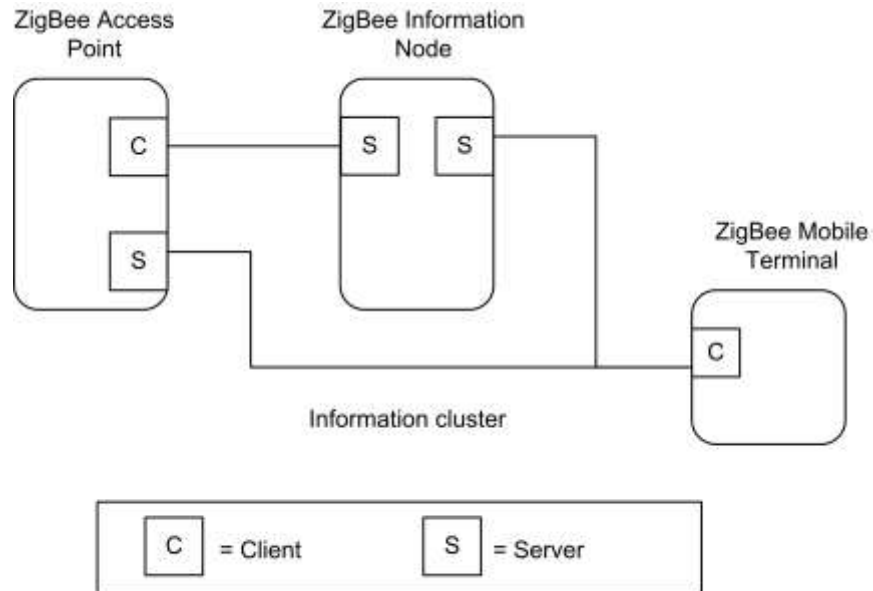
**Table 12-2. Clusters Specified for the Information Delivery**

Cluster Name	Description
Information cluster	Attributes and commands for providing Information service to a ZigBee device.

17601

17602

**Figure 12-2. Typical Usage of the Information Cluster**



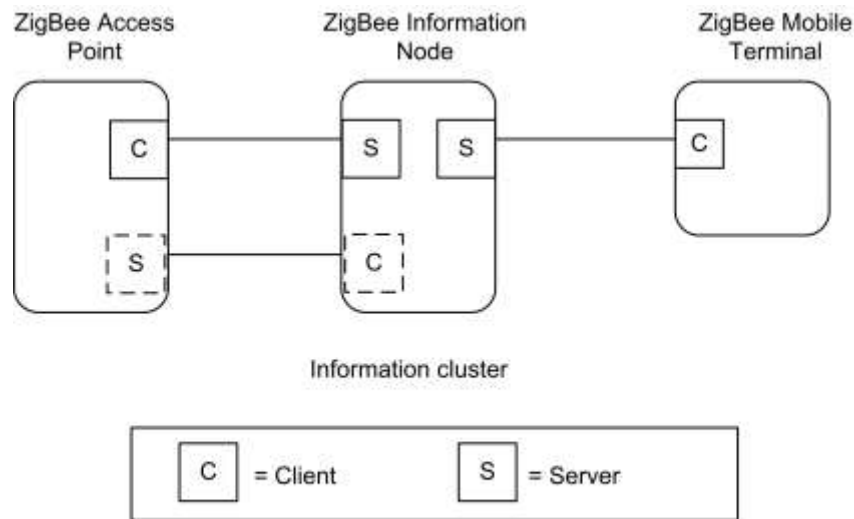
*Note: Device names are examples for illustration purposes only*

17603

17604

17605

**Figure 12-3. Typical Usage of the Information Cluster – with Proxy Function**



*Note: Device names are examples for illustration purposes only  
Note2: Dashed boxes are for the case IN works as proxy for MT*

17606

## 17607 12.2.3 Overview

17608 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
17609 etc.

17610 This cluster provides attributes and commands for Information Delivery Service.

### 17611 12.2.3.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added;CCB 1811 1812 1821

### 17612 12.2.3.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	TELIN	Type 2 (server to client)

### 17613 12.2.3.3 Cluster Identifiers

Identifier	Name
0x0900	Information (Telecom)

## 17614 12.2.4 Server

17615 The Information Node (IN) has a server cluster which provides information delivery service. A client cluster in Mobile  
17616 Terminal (MT) requests information and IN responds with requested contents on pull-based delivery. Besides, cluster  
17617 can provides push-based delivery so the server cluster in the IN sends contents to client cluster in the MT (if properly  
17618 configured).

17619 Content may have links to the other contents. A link is called as child information in this document and it is represented  
 17620 as a ContentID. Contents can be organized in tree-structure.

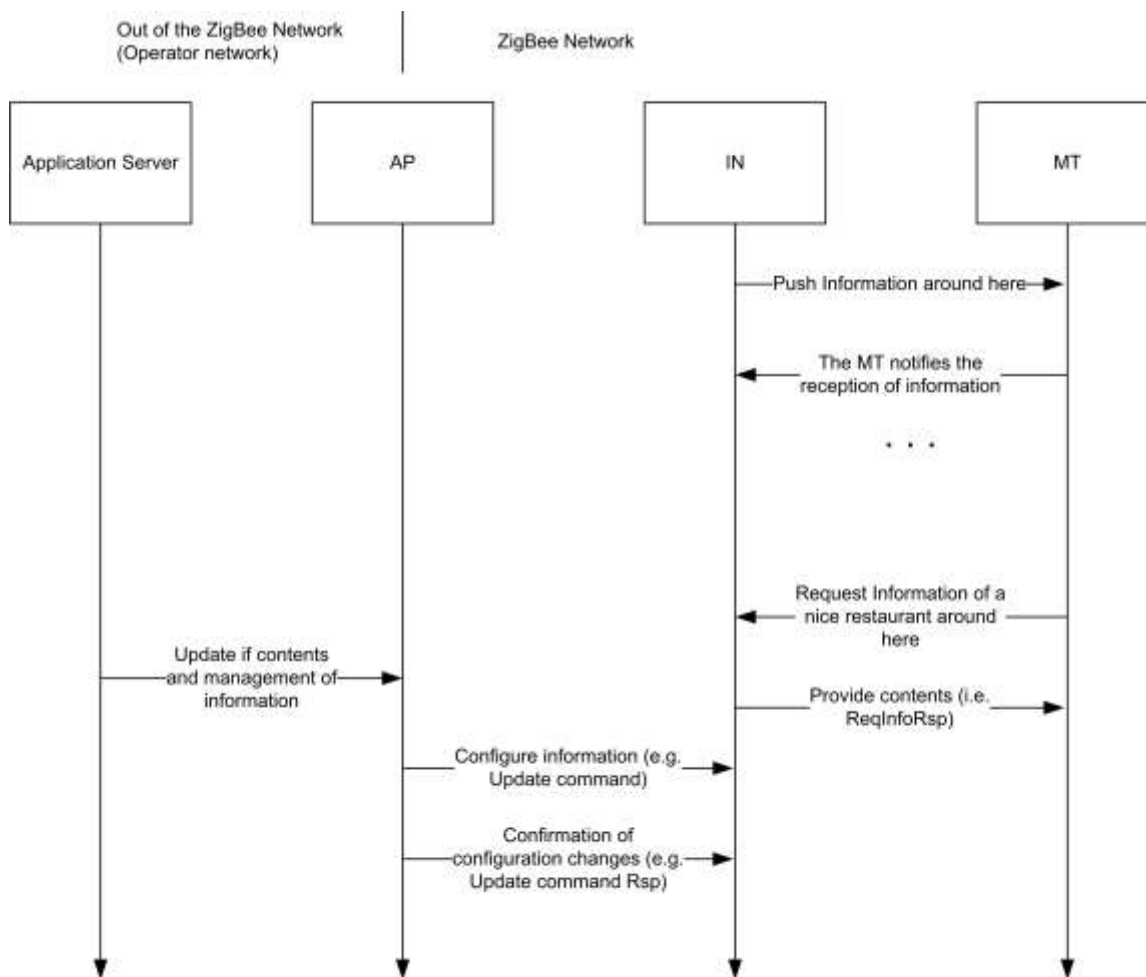
17621 Content may be one of three explicitly specified types: octet strings, character strings or RSS feed, so that the browser  
 17622 in the MT can understand easily what content it should access.

17623 Cluster also provides such function that the client cluster in the AP can update contents and delete them in the IN.

17624 Preference is used for carrying user-side information to let the IN provide user specific contents based on the user-  
 17625 side information. Contents may be modified along with that information on the IN. An example scenario of  
 17626 Information cluster is illustrated in Figure 12-4.

17627 A preference may be processed not in an IN but in an AP or in a server beyond the AP as a gateway. In that case, the  
 17628 IN needs to have client function to forward preference, commands and contents as proxy for MT (Forwarding scenario)  
 17629 or the IN needs to inform the MT to switch its access from the IN to the AP (Redirection scenario). The Cluster  
 17630 supports both scenarios. If the preference, commands and contents are forwarded by the IN between the MT and the  
 17631 AP, they may be just relayed transparently through the IN, or they may be processed by the IN. The IN may process  
 17632 the preference before forwarding it, and may process the stored preference together with the contents to create the  
 17633 customized contents after receiving the response from the AP, then sending the contents to the MT.

17634 **Figure 12-4. An Example Sequence**



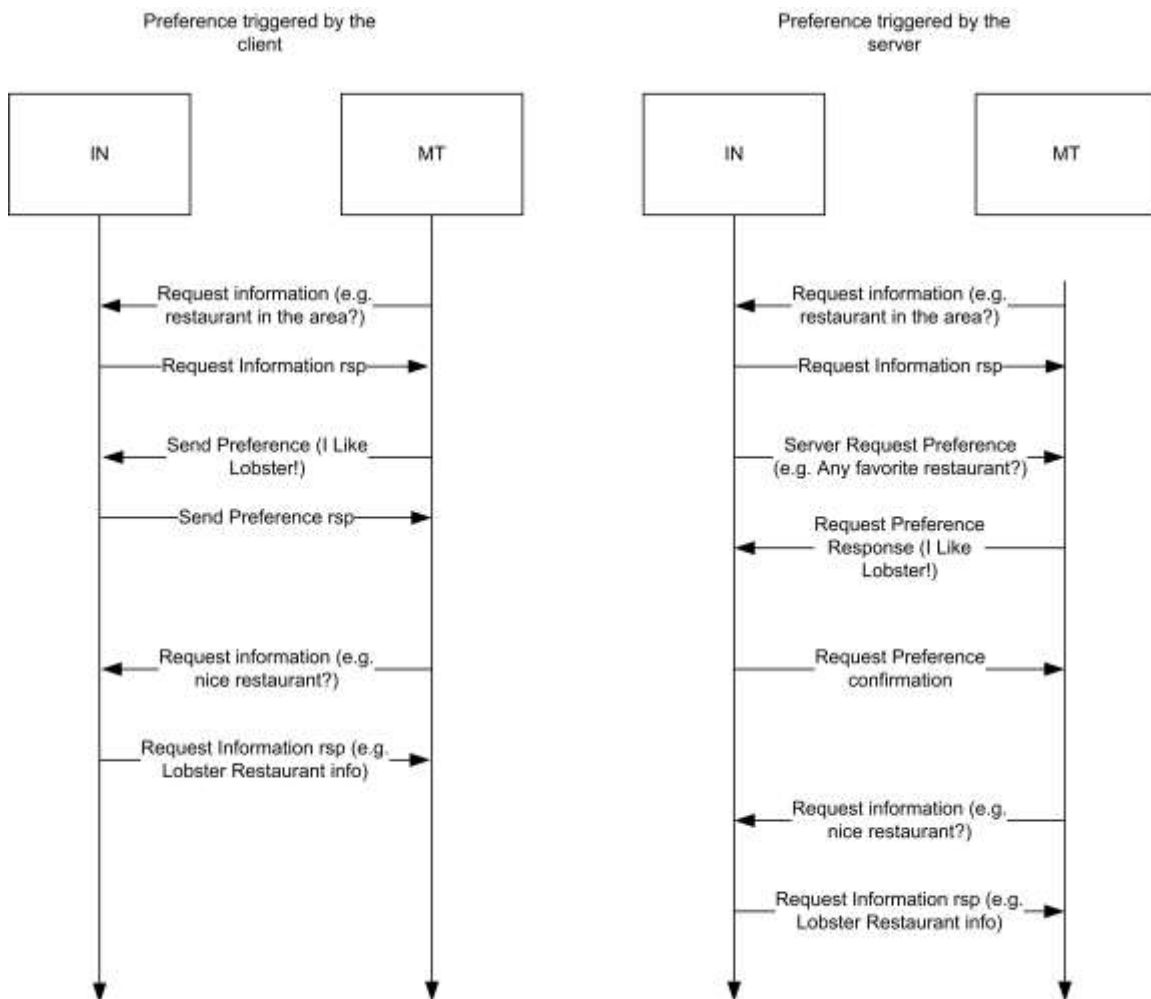
17635  
 17636 **Pull-based service is expected to work as follows:**

- 17637 1. It provides decentralized contents distributed by Update command from the central node (the AP). (e.g.,  
 17638 tree-structure contents distribution, specific permission to peep the contents to the authorized user)

- 17639 2. Advanced application program provides service in conjunction with the other functions like a Location  
17640 cluster, or the preference data from the MT. (e.g., direction service based on the location information of  
17641 user, push service matching individual attribute – invitation of a test drive of new car to men in thirties  
17642 which hobby is driving or etc.)  
17643 3. Hybrid service of the item a. and b.

17644 The preference format is application dependent and is used by the service like the item b. Of course the application  
17645 uses the preference shall have the ability to parse it. If the application doesn't have the ability, it shall report it that.  
17646 The cluster provides a status code to inform it. For example, let's assume the IN provides service like item a. and the  
17647 AP connected a network out of the ZigBee and an application server is deployed there. First the MT access to the IN  
17648 to get general contents for the all of users. The IN can provide simple contents to the MT. Second, the MT request a  
17649 content – good restaurant to the IN, which content is indicated to redirect to the AP. The MT switch its access to the  
17650 AP. The AP requests preference to the MT to get user-side information, favorite food in the example. The AP sends  
17651 the preference to the application server and it replies with the food restaurant which the user likes. Thus the AP can  
17652 provide the content modified along with user-side information – the restaurant which provides he likes – to the MT.  
17653 The example illustrated in Figure 12-5.

17654 **Figure 12-5. Preference Scenarios (Triggered by the Client or by the Server)**



17655

17656 **12.2.4.1 Dependencies**

17657 None



17658 **12.2.4.2 Attributes**

17659 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 17660 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 17661 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 17662 are listed in Table 12-3.

17663 **Table 12-3. Information Cluster Attribute Sets**

Attribute Set Identifier	Description
0x000	Node Information
0x001	Contents Information
0x002 – 0xffff	Reserved

17664 **12.2.4.2.1 Node Information Attribute Set**

17665 The Node Information attribute set contains the attributes summarized in Table 12-4.

17666 **Table 12-4. Node Information Attribute Set**

Id	Name	Type	Range	Access	M/O
0x0000	<i>NodeDescription</i>	string		R	M
0x0001	<i>DeliveryEnable</i>	bool	0x00 – 0x01	R	M
0x0002	<i>PushInformationTimer</i>	uint32		R	O
0x0003	<i>EnableSecureConfiguration</i>	bool	0x00 – 0x01	R	M

17667 **12.2.4.2.1.1 NodeDescription Attribute**

17668 This *NodeDescription* Attribute holds strings which indicate what Information Delivery service is available so that an  
 17669 end-user can select and distinguish this service.

17670 **12.2.4.2.1.2 DeliveryEnable Attribute**

17671 The *Delivery Enable* attribute is Boolean and indicates whether the cluster is able to communicate with the other  
 17672 nodes. It is a read only attribute but it can be changed by using the Configure Delivery Enable command. If it is set to  
 17673 TRUE (0x01), the Information cluster is able to manage the following commands: Request Information, Push  
 17674 Information Response, Send Preference and Request Preference Response.

17675 **12.2.4.2.1.3 PushInformationTimer Attribute**

17676 The *Push Information Timer* is an Unsigned 32-bit integer and indicates whether the cluster is able to send Push  
 17677 Information command and the time between those commands. It is a read only attribute but it can be changed by using  
 17678 the Configure Push Information timer command. If this attribute is set to 0, then the automatic Push Information is  
 17679 disabled, otherwise the value is considered as an interval (in milliseconds) that elapses between Push Information  
 17680 commands. If this attribute is set to 0, it's still possible for the device to push information triggered by an event such  
 17681 as button being pushed.

17682 **12.2.4.2.1.4 EnableSecureConfiguration Attribute**

17683 The Enable Secure Configuration attribute is a Boolean and indicates whether an application layer security is required  
 17684 in order to process the configuration commands: Update, Delete, Configure Delivery Enable, Configure Set Root ID,  
 17685 Configure Node Description, Configure Push Information timer. If this attribute is set to TRUE, then server side of  
 17686 the cluster need to use application link keys for processing those commands. If FALSE, then all the commands can be  
 17687 processed without using link keys.

#### 17688 12.2.4.2.2 Contents Information Attribute Set

17689 The Node Information attribute set contains the attributes summarized in Table 12-5.

17690 **Table 12-5. Contents Information Attribute Set**

Identifier	Name	Type	Range	Access	M/O
0x0010	<i>NumberOfContents</i>	uint16	0x0000 - 0xFFFF	R	O
0x0011	<i>ContentRootID</i>	uint16	0x0000 - 0xFFFF	R	O

##### 17691 12.2.4.2.2.1 NumberOfContents Attribute

17692 This attribute holds the total number of contents which this server node has. It should reflect the result of updating  
 17693 command by AP. This attribute holds the total number of contents which this server node has. If the number is more  
 17694 than 0xffff, this attribute shall be set to 0xffff.

##### 17695 12.2.4.2.2.2 ContentRootID Attribute

17696 This attribute holds root Content ID of octet strings, character string and RSSFeed Contents. *ContentRootID* is a start  
 17697 pointer so that user can access variety contents. If this attribute doesn't exist, there are no contents. 0xffff for this  
 17698 attribute means it is not specified yet.

#### 17699 12.2.4.3 Commands Received

17700 The received command IDs for the information cluster are listed in Table 12-6. Please notice that at least one of the  
 17701 commands shall be implemented though they are defined as optional.

17702 **Table 12-6. Received Command IDs for the Information Cluster**

Id	Description	M/O	Command Type
0x00	Request Information	M	<i>Operation</i>
0x01	Push Information Response	M	<i>Operation</i>
0x02	Send Preference	O	<i>Operation</i>
0x03	Request Preference Response	O	<i>Operation</i>
0x04	Update	O	<i>Configuration</i>
0x05	Delete	O	<i>Configuration</i>
0x06	Configure Node Description	O	<i>Configuration</i>
0x07	Configure Delivery Enable	O	<i>Configuration</i>
0x08	Configure Push Information Timer	O	<i>Configuration</i>
0x09	Configure Set Root ID	O	<i>Configuration</i>

17703 **12.2.4.3.1 Request Information Command**

17704 This is a command requesting information as a list, as a content of text strings and as an RSS feed from mobile terminal  
 17705 to the Information Node or to the Access Point. An Information Node (or an Access Point) that receives this command  
 17706 shall reply by Request Information Response Command with requested information to the sender of this command. It  
 17707 specifies how to indicate content By the ‘Inquiry Type’ and also specifies what data type of content is requested by  
 17708 the ‘Data Type ID’. For example, in pull scenario, MT gets contents list, sending this command (e.g., Inquiry ID =  
 17709 ‘Request by depth’) and receiving Request Information Response Command with the list of titles. By another Request  
 17710 Information Command indicating contents ID, MT can get an individual content.

17711 **12.2.4.3.1.1 Frame Format**

17712 The Request Information command shall be formatted as illustrated in Figure 12-6.

17713 **Figure 12-6. Payload Format of Request Information Command**

<b>Octets</b>	1	1	Variable
<b>Data Type</b>	enum8	map8	See 12.2.4.3.1.2
<b>Field Name</b>	Inquiry ID	Data Type ID	Request Information Payload

17714  
 17715 Inquiry ID shall be set as one of IDs listed in Table 12-7.  
 17716 Data Type ID indicates what type of contents the response command requires. It shall be formatted by combination  
 17717 of bitmasks described in Table 12-8. A bit for ‘Title’ indicates the request requires ‘Title strings’ and it can be  
 17718 combined other type of contents. Flagging ‘Title’ bit means a request title be attached and the other bits used for filter.  
 17719 If ‘Title’ bit, ‘Octet’ bit and ‘RSS’ bit are flagged, that means request is “Octet content attached title and RSS content  
 17720 attached title are required.” In the case that the only ‘Title’ bit is flagged, the request means “Just titles are required.”  
 17721 Please notice that all the contents shall maintain a title in the local database.

17722 **Table 12-7. Inquiry ID**

<b>Inquiry ID</b>	<b>Description</b>	<b>M/O</b>
0x00	Request a content by a content ID	M
0x01	Request contents by multiple IDs	O
0x02	Request all	O
0x03	Request by depth	O

17723  
 17724 **Table 12-8. Data Type IDs**

<b>Data Type ID</b>	<b>Bit Mask</b>	<b>Description</b>
0x01	0000 0001	Title
0x02	0000 0010	Octet String
0x04	0000 0100	Character String

Data Type ID	Bit Mask	Description
0x08	0000 1000	RSS Feed
0x1X – 0xfX	-	Reserved

17725 **12.2.4.3.1.2 Request Information Payload**

17726 Request Information Payload changes along with Inquiry ID listed in Table 12-7. Payload formats for each Inquiry  
17727 ID are described following sections.

17728 **12.2.4.3.1.3 Inquiry ID**

17729 **12.2.4.3.1.3.1 Format for Request a Content by a Content ID**

17730 The command with this ID requests a single content by a Content ID. Format is illustrated in Figure 12-7.

17731 A server shall respond Request Information Response command with a content indicated by the content ID.

17732 **Figure 12-7. Payload Format for Request a Content by a Content ID**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	Content ID

17733 **12.2.4.3.1.3.2 Format for Request Contents by Multiple IDs**

17734 The command with this ID requests several contents by indicating several content IDs. It shall be formatted as  
17735 illustrated in Figure 12-8.

17736 A server shall respond Request Information Response command with contents indicated by content IDs.

17737 **Figure 12-8. Request Information Payload for Request Contents by Multiple IDs**

<b>Octets</b>	2	2	...	2
<b>Data Type</b>	uint16	uint16		uint16
<b>Field Name</b>	Content ID 1	Content ID 2	...	Content ID <i>n</i>

17738 **12.2.4.3.1.3.3 Format for Request All**

17739 The command with this ID requests all contents. No payload format is specified and it should be empty.

17740 A server shall respond Request Information Response command with all contents.

17741 **12.2.4.3.1.3.4 Format for Request by Depth**

17742 Upon receipt of the command with this ID, server shall reply Request Information Response command with  
17743 concatenated contents indicated by Start ID and Depth. Request Information Payload format for this ID is specified in  
17744 Figure 12-9.

- 17745 Start ID field holds content ID for starting point to retrieve structured contents.
- 17746 Depth field holds how many levels to request from Start ID tracing child information. If a depth equals to 0x00, the
- 17747 requested content should be single content of Start ID itself.
- 17748 Server shall provide concatenated contents, which needs a prevention of duplication induced by the loop of links. (For
- 17749 example, if the content has a child content which child ID refers its parent ( $A \rightarrow B$ ,  $B \rightarrow A$ ), there is a loop. If the
- 17750 requester indicates 2 for the depth and requests content "A", searching child information would be like as  $A \rightarrow B \rightarrow A$ .
- 17751 However only content A and B should be carried in this case).

17752 **Figure 12-9. Request Information Payload for Request by Depth**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	Start ID	Depth

17753 **12.2.4.3.2 Push Information Response Command**

- 17754 This command is used by the client to notify the reception of the data carried by Push Information Command, and it
- 17755 is used by the server to confirm if it is correctly stored or not into MT. This command shall not be used if the Push
- 17756 Information Command is sent by broadcast. It is to prevent explosion of response.
- 17757 Payload format shall be as illustrated in Figure 12-10.

17758 **Figure 12-10. Payload Format of Push Information Response Command**

<b>Octets:</b>	2	1	...	2	1
<b>Field:</b>	Notification 1		...	Notification <i>n</i>	
	Content ID 1	Status Feedback 1	...	Content ID <i>n</i>	Status Feedback <i>n</i>

- 17759
- 17760 Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the
- 17761 notification has the status for. Status Feedback indicates the status of the reception of the content.
- 17762 Possible message for Status Feedback are SUCCESS, FAILURE, MALFORMED\_COMMAND,
- 17763 UNSUP\_CLUSTER\_COMMAND, INVALID\_FIELD, INSUFFICIENT\_SPACE, HARDWARE\_FAILURE and
- 17764 SOFTWARE\_FAILURE already specified in the enumeration lists of ZCL [R4].

17765 **12.2.4.3.3 Send Preference Command**

- 17766 This command carries a preference that is specific information of interest for the user, from the client to the server.
- 17767 Upon receipt of this command on the server, the server application may modify or change user specific contents along
- 17768 with preference information. The type of data put into the preference is based on the Preference Type field. Payload
- 17769 format for this command shall be as illustrated in Figure 12-11.

17770

**Figure 12-11. Payload Format for Send Preference Command**

<b>Octet:</b>	2	<b>Variable</b>
<b>Field</b>	Preference Type	Preference Payload, see Table 12-9

17771 The Preference Type determines the format of the preference Payload. All devices must support Preference Type of  
17772 0x0000.

17773

**Table 12-9. Preference Type**

Preference Type	Description
0x0000	Preference is Multiple Content ID
0x0001	Preference is Multiple Octet Strings
0x0002 – 0x7fff	Reserved
0x8000 – 0xffff	Used for Vendor Specific Format
0xfffc – 0xffff	Reserved

17774

17775

**Figure 12-12. Payload Format for Preference Is Multiple Content ID (0x0000)**

<b>Octets</b>	1	2	...	2
<b>Data Type</b>	uint8	uint16	...	uint16
<b>Field Name</b>	Count	Content ID 1		Content ID N (based on Count)

17776

17777

**Figure 12-13. Payload Format for Preference Is Multiple Octet Strings (0x0001)**

<b>Octets</b>	1	Variable (1-256)	...	Variable (1-256)
<b>Data Type</b>	uint8	octstr	...	octstr
<b>Field Name</b>	Count	Preference Data 1		Preference Data N (based on Count)

17778 As described in Figure 12-5 there are two scenarios for the preference:

17779 4. Preference triggered by server side (Information Node or Access Point):

- 17780 • IN ← (Request Information) ← MT
- 17781 • IN → (Request Information Response) → MT
- 17782 • IN → (Server Request Preference) → MT
- 17783 • IN ← (Request Preference Response) ← MT
- 17784 • IN → (Request Preference Confirmation) → MT
- 17785 • IN ← (Request Information) ← MT

- 17786 • IN → (Request Information Response) → MT
- 17787 5. Preference triggered by client side (e.g., Mobile terminal):
- 17788 • IN ← (Request Information) ← MT
- 17789 • IN → (Request Information Response) → MT
- 17790 • IN ← (Send Preference) ← MT
- 17791 • IN → (Send Preference Response) → MT
- 17792 • IN ← (Request Information) ← MT
- 17793 • IN → (Request Information Response) → MT

17794 **12.2.4.3.4 Request Preference Response Command**

17795 This command carries a preference as a response of ‘Server Request Preference’ command on pull-basis. Format shall  
 17796 be as illustrated in Figure 12-14.

17797 **Figure 12-14. Payload Format of Request Preference Response Command**

Octets:	1	2	Variable
Field:	Status Feedback	Preference Type	Preference Payload, see Table 12-9

17798  
 17799 Status Feedback carries a message as a response to previous ‘Server Request Preference’ command. Possible messages  
 17800 are SUCCESS, FAILURE, NOT\_FOUND, MALFORMED\_COMMAND, UNSUP\_CLUSTER\_  
 17801 COMMAND, INVALID\_FIELD, HARDWARE\_FAILURE and SOFTWARE\_FAILURE already specified in  
 17802 enumeration lists of ZCL [R1]. Besides, REQUEST\_DENIED is included to these messages for this cluster  
 17803 specification.

17804 **12.2.4.3.5 Update Command**

17805 Server cluster in the IN which receives this command from the AP shall updates contents by the one which the  
 17806 command carried except that there is an error in the IN. Update command also indicates various control to the contents  
 17807 by the control fields. Control fields affect to all of contents carried by the Update command, so contents required to  
 17808 be indicated different control should be carried by another Update command.

17809 Payload format is as illustrated in Figure 12-15.

17810 **Figure 12-15. Payload Format for Update Command**

Octet	1	1	Variable
Data Type	enum8	map8	Payload Format for Multiple Content
Field Name	Access Control Field	Option Field	Contents Data

17811 **12.2.4.3.5.1 Access Control Field**

17812 Access Control Field is 8-bit enumeration and is used to indicate security level for the validation to access the contents  
 17813 which are carried by the Update command. All of contents carried by the Update command shall be affected by this  
 17814 control field. The enumeration values are listed up in Table 12-10.

17815

**Table 12-10. Value of the Access Control Field**

Access Control Mode Value	Description
0x00	Free to access
0x01	Link key establishment based
0x02	Billing based
0x03 – 0xfe	Reserved
0xff	Vendor Specific

17816 **Free to access:** All of the clients is permitted to access the contents without special validation.

17817 **Link key establishment based:** The client to access to the IN shall be required to establish link key establishment to  
17818 achieve the contents. Contents shall be encrypted by the link key.

17819 **Billing based:** The client to access the contents is required to finish the Billing cluster procedure.

17820 **Vendor specific:** No special method is defined in this document. The application defines it. (Out-of-box, out-of-band,  
17821 etc.)

17822 **12.2.4.3.5.2 Option Field**

17823 Option Field is used for advanced indication while updating contents. Forwarding flag, Redirection flag, Overwrite  
17824 update flag are defined in the current version. The ‘Forwarding’ flag or the ‘Redirection’ flag are used to indicate  
17825 ‘content’ so that the commands of request and response related to the indicated ‘content’ shall be forwarded or  
17826 redirected to the AP. If both ‘Forward’ flag and ‘Redirection’ flag are 1, the server cluster shall reply the  
17827 INVALID\_FIELD by the Update Response command.

17828 The format is as illustrated in Figure 12-16.

17829 **Figure 12-16. Format for Redirection Control Field**

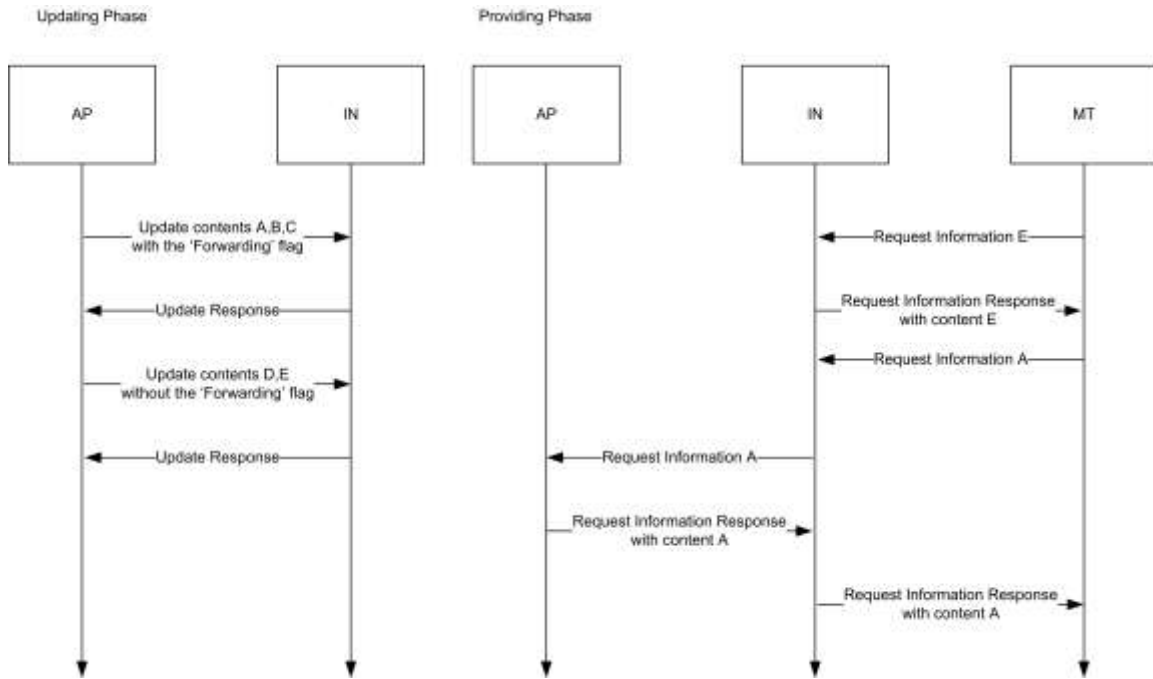
<b>Bits: 1</b>	<b>1</b>	<b>1</b>	<b>5</b>
Forward	Redirection	Overwrite update	Reserved

17830 **Forward flag:** The Information Node is required to forward messages from the MT to the AP and message from the  
17831 AP to the MT with acting as proxy. All the requests from the MT for the contents updated with this flag are forwarded  
17832 to the AP. A Preference from the MT is also sent to AP if the IN has it. The AP answers the response command to the  
17833 IN with requested contents and they are forwarded to the MT similarly. Figure 12-17 shows an example usage of  
17834 forwarding.



17835

Figure 12-17. An Example Sequence of Forwarding Case



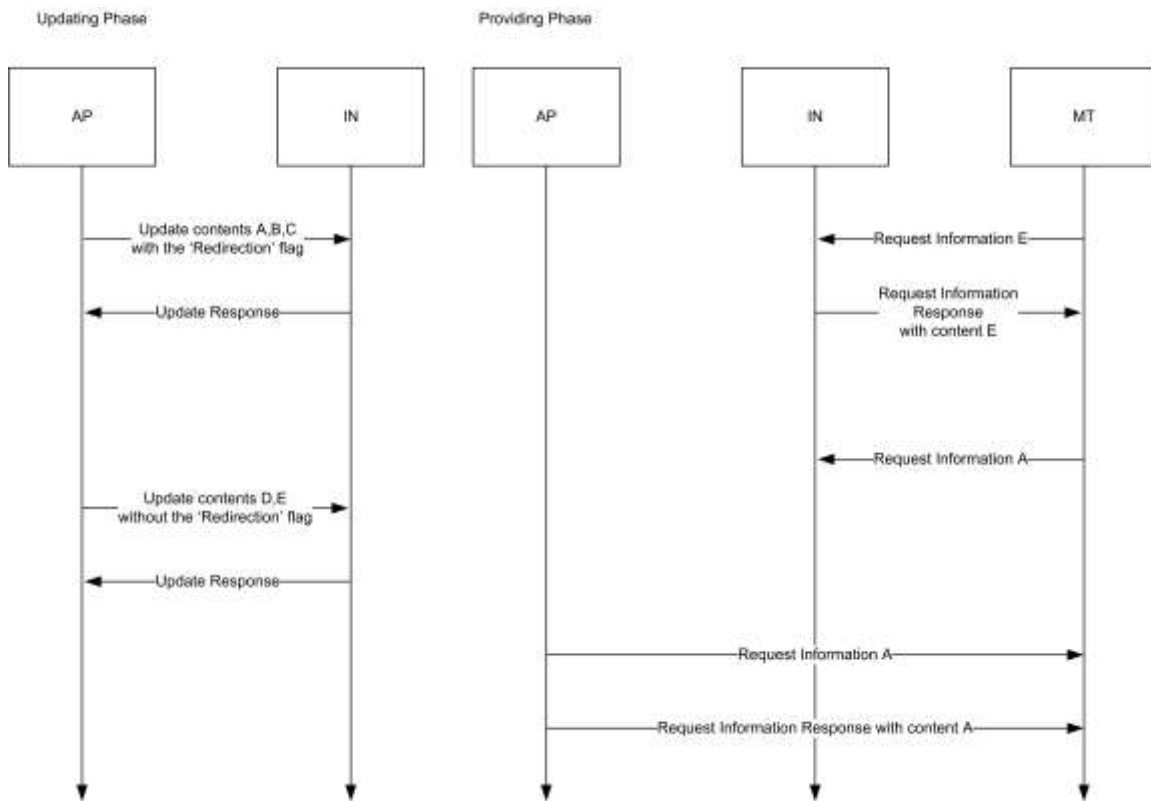
17836

17837

17838 **Redirection flag:** A client requested the contents indicated by this flag shall receive Request Information Response  
 17839 command with Status feedback 'INDICATION\_REDIRECTION\_TO\_AP'. The client is required to switch to access  
 17840 from the Information Node to the Access Point. This flag makes MT enable to switch access to AP automatically  
 17841 without user's operation (Like that user access the child content). For example, let IN have general site-dependent  
 17842 information and content depends on user-side information generated by a server beyond the operator network which  
 17843 AP is connected. Figure 12-18 shows an example usage of redirecting.

17844

Figure 12-18. An Example Sequence of Redirecting Case



17845

17846

17847 **Overwrite:** For the case that the IN has already contents corresponding to the one required to Update, the command  
 17848 indicates if it can be overwritten or not. If it is 0b1, the contents carried by the Update command overwrites the  
 17849 contents which the IN has. If it is 0b0, overwriting is not permitted. In that case, the error 'FAILURE' on Update  
 17850 Response command is issued by the IN and the command is ignored if the IN has corresponding contents.

17851 **12.2.4.3.6 Delete Command**

17852 Server cluster in the IN which receives this command from the AP shall delete contents by the one which the command  
 17853 carried except that there is an error in the IN. Delete command also indicates various control to the contents by the  
 17854 control fields. Control fields affect to all of contents carried by the Delete command, so contents required to be  
 17855 indicated different control should be carried by another Delete command.

17856 Payload format is as illustrated in Figure 12-19.

17857

Figure 12-19. Payload Format for Delete Command

<b>Octet</b>	1	2	...	2
<b>Data Type</b>	map8	uint16	...	uint16
<b>Field Name</b>	Deletion Option	ContentID 1	...	ContentID <i>n</i>

17858 **12.2.4.3.6.1 Deletion Option Field**

17859 Deletion Option field enables various deletion functions. The format is as illustrated in Figure 12-20.

17860 **Figure 12-20. Format for Deletion Option Field**

<b>Bits: 1</b>	<b>2-8</b>
Recursive	Reserved

17861  
 17862 **Recursive:** If it is 0b1, all the sub tree starting for the content carried by the Delete command are deleted. If it is 0b0,  
 17863 only the content carried by the Delete command is deleted. How the children are linked to the rest of the tree is out of  
 17864 scope of this document, it is application dependent.

17865 **12.2.4.3.7 Configure Node Description**

17866 Payload format for the Configure Node Description command shall be as illustrated in Figure 12-21.

17867 Upon recipient of this command, the server cluster shall change its Node Description attribute to the value of the  
 17868 “Description” field in this command. The use of this specific command guarantees that Node Description attribute can  
 17869 be reconfigured only when Delivery Enable attribute is set to TRUE. Upon reception of this command the recipient  
 17870 will reply with Default Response with status field equal to SUCCESS (if the requester set the Disable Default response  
 17871 bit of ZCL header to 0). The Configure Node Description command will be acknowledged with Default Response  
 17872 with status field equal to NOT\_AUTHORIZED in case the recipient entity requires a secure link for configuration  
 17873 (i.e., Enable Secure Configuration attribute set to TRUE) while the sender didn’t use the proper link key for sending  
 17874 the configuration commands.

17875 **Figure 12-21. Payload Format for Configure Node Description Command**

<b>Octets</b>	<b>Variable</b>
<b>Data Type</b>	string
<b>Field Name</b>	Description

17876 **12.2.4.3.8 Configure Delivery Enable**

17877 Payload format for the Configure Delivery Enable command shall be as illustrated in Figure 12-22.

17878 Upon recipient of this command, the server side of the Information cluster should set the value present in the ‘Enable  
 17879 flag’ field into the Delivery Enable attribute. Note that, if Enable Secure Configuration attribute is set to TRUE (0x01)  
 17880 this command should be handled only whether the entity sending this message will have previously set up its link key  
 17881 with the recipient entity.

17882 If the ‘Enable flag’ is set to FALSE (0x00), the server cluster shall stop the information delivery service. However the  
 17883 device supporting this server cluster (i.e., Information node) shall still accept those commands needed to configure  
 17884 the node: Update, Delete, Configure Node Description, Configure Delivery Enable, Configure Push Information timer  
 17885 and Configure Set Root ID.

17886 All cluster specific commands except from “configuration” commands will be replied by their respective responses  
 17887 with status code REQUEST\_DENIED if the Delivery Enable attribute is disabled.

17888 The Configure Delivery Enable command will be acknowledged with Default Response with status field equal to  
 17889 NOT\_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure  
 17890 Configuration attribute set to TRUE) while the sender didn’t use the proper link key for sending the configuration  
 17891 commands.

17892

**Figure 12-22. Payload Format for Configure Delivery Enable Command**

<b>Octets:</b>	<b>1</b>
<b>Field:</b>	Enable flag

17893 **12.2.4.3.9 Configure Push Information Timer**

17894 Payload format for the Configure Push Information Timer command shall be as illustrated in Figure 12-23.

17895 Upon recipient of this command, the server side of the Information cluster shall change its Push Information timer  
17896 attribute to the value of the ‘Timer’ field carried in this command.17897 The Configure Push Information Timer command will be acknowledged with Default Response with status field equal  
17898 to NOT\_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure  
17899 Configuration attribute set to TRUE) while the sender didn’t use the proper link key for sending the configuration  
17900 commands.

17901

**Figure 12-23. Payload Format for Configure Push Information Timer Command**

<b>Octets:</b>	<b>4</b>
<b>Field:</b>	Timer

17902 **12.2.4.3.10 Configure Set Root ID**

17903 Payload format for the Configuration Set Root ID command shall be as illustrated in Figure 12-24.

17904 Upon recipient of this command, the server side of Information cluster shall change its Root ID attribute to the value  
17905 of the ‘Root ID’ field in this command.17906 The Configure Set Root ID command will be acknowledged with Default Response with status field equal to  
17907 NOT\_AUTHORIZED in case the recipient entity requires a secure link for configuration (i.e., Enable Secure  
17908 Configuration attribute set to TRUE) while the sender didn’t use the proper link key for sending the configuration  
17909 commands.

17910

**Figure 12-24. Payload Format for Configure Set Root ID Command**

<b>Octets:</b>	<b>2</b>
<b>Field:</b>	Root ID

17911 **12.2.4.4 Commands Generated**17912 The generated command IDs for the Information cluster are listed in Table 12-11. Please notice that at least one of the  
17913 following commands shall be implemented.

17914

**Table 12-11. Generated Command IDs for the Information Cluster**

Command Identifier Field Value	Description	M/O	Command Type
0x00	Request Information Response	M	<i>Operation</i>
0x01	Push Information	M	<i>Operation</i>

Command Identifier Field Value	Description	M/O	Command Type
0x02	Send Preference Response	O	<i>Operation</i>
0x03	Server Request Preference	O	<i>Operation</i>
0x04	Request Preference Confirmation	O	<i>Operation</i>
0x05	Update Response	O	<i>Configuration</i>
0x06	Delete Response	O	<i>Configuration</i>

17915 **12.2.4.4.1 Request Information Response Command**

17916 This command is a response command according to a Request Information command which a client requests and  
 17917 carries requested information or carries status feedback if error occurs. Payload format for this command shall be as  
 17918 illustrated in Figure 12-25.

17919 **Figure 12-25. Payload Format of Request Information Response Command**

<b>Octet:</b>	<b>1</b>	<b>1</b>	<b>Variable</b>	<b>...</b>	<b>1</b>	<b>Variable</b>
<b>Field:</b>	Number	Status Feedback 1	Single Content or ContentID	...	Status Feedback <i>n</i>	Single Content or ContentID

17920  
 17921 Number indicates how many single contents are carried by this command. Pairs of ‘Status Feedback’ and ‘Single  
 17922 Content’ appear corresponding to this number.

17923 Single content is the actual content which format is specified in 12.2.6.1.

17924 Status Feedback carries a message as a response to the previous ‘Request Information’ command sent by the client.  
 17925 Possible messages are SUCCESS, FAILURE, NOT\_FOUND, MALFORMED\_COMMAND,  
 17926 UNSUP\_CLUSTER\_COMMAND, INVALID\_FIELD, HARDWARE\_FAILURE and SOFTWARE\_FAILURE  
 17927 already specified in enumeration list. Besides, INDICATION\_REDIRECTION\_TO\_AP, REQUEST\_DENIED,  
 17928 PREFERENCE\_IGNORED and MULTIPLE\_REQUEST\_NOT\_ALLOWED are included to these messages for this  
 17929 cluster specification. All the Status enumerations are listed in Table 12-12.

17930 If a content is not available due to some reason (an error in many cases), the ‘Single Content’ field should be replaced  
 17931 by the “Content ID” in order to report which content requested has an error. (i.e., all the status codes except SUCCESS  
 17932 and PREFERENCE\_IGNORED).

17933 **12.2.4.4.2 Push Information Command**

17934 This is a command putting information especially from an information node (or an access point) to a mobile terminal  
 17935 on push basis. A content sent to mobile terminal (e.g., a list of contents, a content described in octets strings, character  
 17936 strings, a title of content or RSS feed) is carried by this command.

17937 **Figure 12-26. Payload Format of Push Information Command**

<b>Octet:</b>	<b>Variable</b>
<b>Field:</b>	Contents Data

17938

17939 Format for Contents Data shall be as illustrated in 12.2.6.

#### 17940 **12.2.4.4.3 Send Preference Response Command**

17941 This command is used by the server to notify whether the data carried by Send Preference Command generated by the  
17942 client is accepted correctly or not.

17943 Payload format shall be as illustrated in Figure 12-27.

17944 Status Feedback carries a message as a response to the previous command ‘Send Preference’ command from the client.  
17945 Possible values are: SUCCESS, ZCL\_PREFERENCE\_DENIED, ZCL\_PREFERENCE\_IGNORED. If all the  
17946 Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to SUCCESS.

17947 Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback  
17948 value will be set to ZCL\_PREFERENCE\_IGNORED (0x74).

17949 **Figure 12-27. Payload Format for Send Preference Response Command and Request Preference Confirmation Command**

<b>Octet:</b>	<b>1</b>	<b>...</b>	<b>1</b>
<b>Field:</b>	Status Feedback 1	...	Status Feedback <i>n</i>

#### 17950 **12.2.4.4.4 Server Request Preference Command**

17951 This command requests a Preference as user-side information in the MT on pull-basis.

17952 Upon receipt of this command at client cluster in the MT, the client is required to respond by Request Preference  
17953 Response command.

17954 This command has no payload.

#### 17955 **12.2.4.4.5 Request Preference Confirmation Command**

17956 This command is used by the server to notify whether the data carried by Request Preference Response command  
17957 generated by the client is accepted correctly or not.

17958 Payload format shall be as illustrated in Figure 12-27 above.

17959 Status Feedback carries a message as a response to the previous command ‘Request Preference Response’ command  
17960 from the client. Possible values are: SUCCESS, ZCL\_PREFERENCE\_DENIED, ZCL\_PREFERENCE\_IGNORED.  
17961 If all the Preference Data are correctly processed, it is enough to respond with a unique Status Feedback equals to  
17962 SUCCESS.

17963 Also, if the server device does not support the Preference Type carried by the command, a unique Status Feedback  
17964 value will be set to ZCL\_PREFERENCE\_IGNORED (0x74).

#### 17965 **12.2.4.4.6 Update Response Command**

17966 This command is used to notify any result of Update Command received by IN.

17967 Payload format for this command shall be as illustrated in Figure 12-28.

17968 Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the  
17969 notification has the status for. Status Feedback indicates the status of the reception of the content.

17970

**Figure 12-28. Payload Format of Update Response and Delete Response command**

<b>Octet:</b>	<b>2</b>	<b>1</b>	<b>...</b>	<b>2</b>	<b>1</b>
<b>Field:</b>	Notification 1		...	Notification <i>n</i>	
	Content ID 1	Status Feedback 1		Content ID <i>n</i>	Status Feedback <i>n</i>

17971

17972 Status Feedback carries a message as a response to the previous command ‘Update’ command from the client. Possible  
 17973 messages are SUCCESS, FAILURE, MALFORMED\_COMMAND, UNSUP\_CLUSTER\_  
 17974 COMMAND, INVALID\_FIELD, INSUFFICIENT\_SPACE, HARDWARE\_FAILURE and  
 17975 SOFTWARE\_FAILURE already specified in enumeration lists of ZCL [R1]. Besides, REQUEST\_DENIED is  
 17976 included into these messages for this cluster specification. All the status enumerations are listed in Table 12-12.

17977 **12.2.4.4.7 Delete Response Command**

17978 This command is used to notify any result of Delete Command received by IN.

17979 Payload format for this command shall be as illustrated in Figure 12-28.

17980 Notification field has two sub-fields, content ID and Status Feedback. Content ID indicates what content the  
 17981 notification has the status for. Status Feedback indicates the status of the reception of the content.

17982 **12.2.5 Client**

17983 **12.2.5.1 Command Received**

17984 The client receives the cluster specific commands detailed in 12.2.4.4.

17985 **12.2.5.2 Command Generated**

17986 The client generates the cluster specific commands detailed in 12.2.4.3, as required by application.

17987 **12.2.6 Payload Formats for Contents Data**

17988 This section describes about payload format for contents data as used in the commands defied for the Information  
 17989 cluster.

17990 **12.2.6.1 Payload Format for Multiple Contents**

17991 Payload format for the contents shall be as illustrated in Figure 12-29.

17992

**Figure 12-29. Payload Format for Multiple Contents**

<b>Octet</b>	1	Variable	...	Variable
<b>Data Type</b>	uint8	Format for Single Content (defined in this section)		Format for Single Content (defined in this section)
<b>Field Name</b>	Number	Single Content 1	...	Single Content <i>n</i>

17993

17994 Number field holds a number of single contents. The payload format for the single content is specified in Figure 12-30.

17995

**Figure 12-30. Format for Single Content**

<b>Octet</b>	2	1	Variable	Variable	1	2/0	...	2/0
<b>Data Type</b>	uint16	map8	Long Character String (defined in this cluster section)	Payload Format for 'Content' (defined in this cluster section)	uint8	uint16	....	uint16
<b>Field Name</b>	Content ID	Data Type ID	Title String	Content String	Number of children	Content ID 1	...	Content ID <i>n</i>

17996

17997 Content ID corresponds to the content. There is no rule provided by this document. It is expected to be defined by the  
17998 service provider.

17999 Data Type indicates the supported data types of content (it could be title and/or long octet, long character string or  
18000 RSS). If a combination of type is supported by a Single Content, the order of data types shall be the one described  
18001 in Figure 12-30. If a bit field of 'Title' in Data Type ID is 0b1, 'Title String' field will be inserted in the Single Content  
18002 frame. If another bit than the 'Title' field is 0b1, 'content strings' field and following fields appear.

18003 Title String appears in the Single Content frame only if a 'Title' bit in Data Type ID field is flagged. It represents title  
18004 string in 'character string' data type; 'long character string' data type already includes 2 bytes count field.

18005 Content String holds actual content data described in data type. It is inserted in the frame only if another bit than the  
18006 'Title' field is 0b1 in the Data Type ID field.

18007 Number of Children indicates how many links to child-contents this content has. If there is no child for this content  
18008 this field shall be set to 0.

18009 Content ID *n* holds List of child-contents ID.



18010 **12.2.6.2 Contents Data Types**

18011 **12.2.6.2.1 Title String**

18012 **Figure 12-31. Format for Title String**

<b>Octet: 2</b>	<b>Variable</b>
Count	Title

18013 **12.2.6.2.2 Long Octet String**

18014 Extended count field to two bytes. Count represents how many octets the Octet Data’s length is.

18015 **Figure 12-32. Format for Long Octet String**

<b>Octet: 2</b>	<b>Variable</b>
Count	Octet Data

18016 **12.2.6.2.3 Long Character String**

18017 Extended count field to two bytes. Count represents how many characters the Character Data’s length is. It should not  
 18018 be in Bytes if the character set is not 8-bit code (e.g., 2-bytes code).

18019 **Figure 12-33. Format for Long Character String**

<b>Octet: 2</b>	<b>Variable</b>
Count	Character Data

18020 **12.2.6.2.4 RSS Feed**

18021 Length field represents length in bytes not in character count. What character set is used should be defined in RSS  
 18022 feed data. In many cases, it would be ASCII compatible coding – like a UTF-8.

18023 **Figure 12-34. Format for RSS Feed**

<b>Octet: 2</b>	<b>Variable</b>
Length	RSS Feed Data

18024 **12.2.6.3 Status Codes for the Information Cluster**

18025 Where an information cluster command contains a status field, the actual value of the enumerated status values are  
 18026 listed in Table 12-12. Because this table copied from ZCL status code enumeration and is inserted the Information  
 18027 cluster specific status codes at the start point, it may differ from the original if ZCL is updated. However, there is no  
 18028 problem because this table is used only for information cluster and is only used by its specific commands.

18029

**Table 12-12. Enumerated Status Values Used in the ZCL**

Enumerated status	Value	Description
SUCCESS	0x00	Operation was successful.
FAILURE	0x01	Operation was not successful.
-	0x02 – 0x6f	Reserved.
REQUEST_DENIED	0x70	Request was denied due to lack of permission
MULTIPLE_REQUEST_NOT_ALLOWED	0x71	Request of multiple contents is not supported
INDICATION_REDIRECTION_TO_AP	0x72	Server Indicates to change the access to the AP for this content.
PREFERENCE_DENIED	0x73	The preference was not accepted. Because it was not understandable or invalid.
PREFERENCE_IGNORED	0x74	Inform that preference was not used to modify the content which is provided by this command.
MALFORMED_COMMAND	0x80	The command appears to contain the wrong fields, as detected either by the presence of one or more invalid field entries or by there being missing fields. Command not carried out. Implementer has discretion as to whether to return this error or INVALID_FIELD.
UNSUP_CLUSTER_COMMAND	0x81	The specified general ZCL command is not supported on the device. Command not carried out.
INVALID_FIELD	0x85	At least one field of the command contains an incorrect value, according to the specification the device is implemented to. Command not carried out.
INSUFFICIENT_SPACE	0x89	An attempt to create an entry in a table failed due to an insufficient amount of free space available.
NOT_FOUND	0x8b	The requested information (e.g., table entry) could not be found.
-	0x8d – 0xbf	Reserved
HARDWARE_FAILURE	0xc0	An operation was unsuccessful due to a hardware failure.
SOFTWARE_FAILURE	0xc1	An operation was unsuccessful due to a software failure.
-	0xc3 – 0xff	Reserved

## 18030 **12.3 Chatting**

---

### 18031 **12.3.1 Introduction**

18032 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
18033 etc.

#### 18034 **12.3.1.1 Scope and Purpose**

18035 This section specifies the Chatting cluster, which provides commands and attributes for sending chat messages among  
18036 ZigBee devices. This cluster is to provide a standardized interface for people using ZigBee devices to chat with each  
18037 other like they using instant messaging applications through Internet. The transaction sequence numbers used in the  
18038 ZCL command frames for the Chatting cluster should be the same for the requests and responses; the default responses  
18039 should use also the same transaction sequence numbers of the related commands in order to match the correspondent  
18040 packets.

18041 There are two kinds of chatting scenarios:

##### 18042 6. Centralized Server

18043 In this kind of scenario a centralized server is used for managing and controlling the messaging between the  
18044 different ZigBee nodes. Different chat sessions can be made available by the server. The node entering the  
18045 ZigBee network may search for the available chat sessions and join one of them after choosing one out of  
18046 different available sessions. Different nodes can join one chat session and can interact with each other.

##### 18047 7. Ad Hoc Chat Sessions

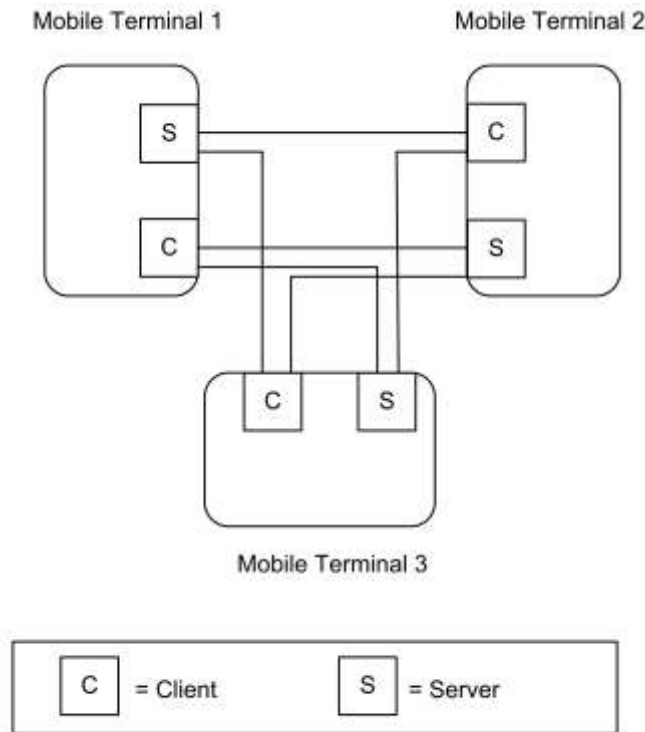
18048 In this kind of scenario no infrastructure is needed. Any ZigBee node can start and manage a chat session. A  
18049 ZigBee node in a particular ZigBee network can start a chat session. A node should only be a chairman of one  
18050 chat session, i.e., it should only start one chat session. It is recommended to do so, since in the ad hoc scenario,  
18051 the chairman can be any devices which may have low computing power and capability, and maintaining more  
18052 than one session may be difficult for the devices. To start a chat session it has to decide a unique identifier for  
18053 the chat session. This identifier shall be unique among all the chat sessions in the networks. For this requirement  
18054 the implementer shall make it mandatory for a node to select a chat identifier which will be unique in the whole  
18055 ZigBee network. The identifier may be set the same as the address of the device that starts the session, so as to  
18056 guarantee its uniqueness.

18057 This document should be used in conjunction with Chapter 2, Foundation, which gives an overview of the library and  
18058 specifies the frame formats and general commands used therein.

18059 This cluster provides attributes and commands for devices to send chatting messages to each other.

18060

**Figure 12-35. Typical Usage of the Chatting Cluster**



*Note: Device names are examples for illustration purposes only*

18061

18062 **12.3.1.2 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

18063 **12.3.1.3 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	CHAT	Type 1 (client to server)

18064 **12.3.1.4 Cluster Identifiers**

Identifier	Name
0x0905	Chatting

18065 **12.3.2 Server**

18066 The server manages the list of participants, the chat session ID, etc. It can respond to the devices which are going to  
 18067 join chat sessions, or it can ask someone to leave the chat session. The server has the functions which are more related  
 18068 to managing the session than sending chatting messages.

18069 The messages in the chat sessions are usually sent by the multicast method. So the server should also manage the chat  
 18070 group. There is an example which can be a guideline for implementing. Once the server forms a new chat session, it  
 18071 should form a new group. The group ID may be the same as the chat session ID. If a new user joins the chat session,  
 18072 it should also join the chat group. To fulfill this, the server should use the Add Group command specified in groups  
 18073 cluster to add the newcomer to the chat group. And if a user leaves the chat session, it should also leave the chat group.  
 18074 To fulfill this, the server should use the Remove Group command specified in groups cluster to remove the user from  
 18075 the chat group.

### 18076 **12.3.2.1 Dependencies**

18077 This cluster does not depend on any other existing clusters. However, in order to successfully fulfill the chatting,  
 18078 Information cluster, Groups cluster and Billing cluster may also need to be implemented in the same device where the  
 18079 chatting cluster is implemented.

### 18080 **12.3.2.2 Attributes**

18081 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 18082 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 18083 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 18084 are listed in Table 12-13.

18085 **Table 12-13. Chatting Attributes Sets**

Attribute Set Identifier	Description
0x000	User Related
0x001	Chat Session Related

#### 18086 **12.3.2.2.1 User Related Attribute Set**

18087 The User Related Attribute Set contains the attributes summarized in Table 12-14.

18088 **Table 12-14. Attributes of the User Related Attribute Set**

Identifier	Name	Type	Range	Access	M/O
0x0000	<i>U_ID</i>	uint16	0x0000-0xffff	R	M
0x0001	<i>Nickname</i>	string		R	M

##### 18089 **12.3.2.2.1.1 U\_ID Attribute**

18090 The *U\_ID* attribute is the unique identification of the user in the chat room. It may be same as the address given to the  
 18091 device while joining in the ZigBee network, or may be same as the 2 least significant bytes of UserID of Billing  
 18092 cluster. The value 0xffff means this attribute is not set.

##### 18093 **12.3.2.2.1.2 Nickname Attribute**

18094 The *Nickname* attribute is a unique display name of the user identified by the *U\_ID* while talking in the public chat  
 18095 room. User sets the *Nickname* while joining the chat room.

#### 18096 **12.3.2.2.2 Chat Session Related Attribute Set**

18097 The Chat Session Related set contains the attributes summarized in Table 12-15.

18098

**Table 12-15. Attributes of Chat Session Related Attribute Set**

Identifier	Name	Type	Range	Access	M/O
0x0010	<i>C_ID</i>	uint16	0x0000-0xffff	R	M
0x0011	<i>Name</i>	string		R	M
0x0012	<i>EnableAddChat</i>	bool	TRUE/FALSE	R	O

18099 **12.3.2.2.2.1 C\_ID Attribute**

18100 The *C\_ID* attribute is the unique identification of a chat room. It is assigned by the chat server while creating a new  
 18101 chat room following the user command or chosen by the chairman. It may be same as the chat group ID. If the server  
 18102 maintains several chat rooms, this attribute should be set as the ID of the latest formed chat room. The value 0xffff  
 18103 means this attribute is not set.

18104 **12.3.2.2.2.2 Name Attribute**

18105 The *Name* attribute is the name or topic of the chat room which is identified by the *C\_ID* attribute.

18106 **12.3.2.2.2.3 EnableAddChat Attribute**

18107 The *EnableAddChat* attribute indicates whether the server permits other users to add new chat rooms in it. TRUE  
 18108 (0x01) indicates the server permit other users to add new chat rooms, while FALSE (0x00) indicates not permit to do  
 18109 so.

18110 **12.3.2.3 Commands Received**

18111 The received commands IDs for the chatting cluster are listed in Table 12-16.

18112

**Table 12-16. Command IDs for the Chatting Cluster**

Command Identifier Field Value	Description	M/O
0x00	Join Chat Request	M
0x01	Leave Chat Request	M
0x02	Search Chat Request	M
0x03	Switch Chairman Response	O
0x04	Start Chat Request	O
0x05	ChatMessage	M
0x06	Get Node Information Request	O

18113 **12.3.2.3.1 Join Chat Request Command**

18114 The Join Chat Request command is used for a node to request to join one chatting session.

18115 The Join Chat request command shall be formatted as illustrated in Figure 12-36.

18116

**Figure 12-36. Format of the Join Chat Request Command**

<b>Octets</b>	2	Variable	2
<b>Data Type</b>	uint16	string	uint16
<b>Field Name</b>	U_ID	Nickname	C_ID

18117

The U\_ID field indicates unique identification of the user in the chat room.

18118

The Nickname field is type of character string which is a unique display name of the user while talking in the public chat room.

18119

18120

The C\_ID field is unique identification of a chat room. It indicates the ID of the chat room which the client wants to join.

18121

18122

This command should be unicast to the server which manages the chat room indicated by the C\_ID.

18123

### **12.3.2.3.2 Leave Chat Request Command**

18124

The Leave Chat Request command is used for a node to request to leave one chatting session. The client may require the Default Response command to be sent from the server so that to confirm the leave command has been successfully received.

18125

18126

18127

The Leave Chat request command shall be formatted as illustrated in Figure 12-37.

18128

**Figure 12-37. Format of the Leave Chat Request Command**

<b>Octets</b>	2	2
<b>Data Type</b>	uint16	uint16
<b>Field Name</b>	C_ID	U_ID

18129

18130

The C\_ID field indicates the ID of the chat room which the node wants to leave. The U\_ID field indicates unique identification of the user in the chat room.

18131

18132

This command should be unicast to the server which manages the chat room the user to leave.

18133

### **12.3.2.3.3 Search Chat Request Command**

18134

The Search Chat Request command is used for a node to request to search for the available chat session on the server.

18135

The Search Chat Request command shall contain no payload and shall be originated by the devices which want to have a chat with others and sent to the server. It may be broadcast in the network.

18136

18137

### **12.3.2.3.4 Switch Chairman Response Command**

18138

The Switch Chairman Response command is used for nodes to response to the Switch Chairman Request command.

18139

The Switch Chairman Response command shall be formatted as illustrated in Figure 12-38.

18140

**Figure 12-38. Format of the Switch Chairman Response Command**

<b>Octets</b>	2	2
<b>Data Type</b>	uint16	uint16
<b>Field Name</b>	C_ID	U_ID

18141

18142 The C\_ID field in the command indicates the ID of the chat room of which the receiving node is the old chairman.  
18143 The U\_ID field in the command is the unique ID of node which wants to be the chairman of the chat room.

18144 This command shall be unicast to the chairman, announcing that the node indicated by the U\_ID volunteers to be the  
18145 new chairman.

### 18146 12.3.2.3.5 Start Chat Request Command

18147 The Start Chat Request command is used for a device to request to create one chat session. The new chat session to  
18148 be created shall be managed by the responder. That is, once the chat session is created, the responder but not the  
18149 requester will be the chairman of the chat room.

18150 The Start Chat request command shall be formatted as illustrated in Figure 12-39.

18151

**Figure 12-39. Format of the Start Chat Request Command**

<b>Octets</b>	Variable	2	Variable
<b>Data Type</b>	string	uint16	string
<b>Field Name</b>	Name	U_ID	Nickname

18152 The Name field indicates the topic of the chat room. The U\_ID field indicates unique identification of the user in the  
18153 chat room. The Nickname field indicates the Nickname set by the requester.

18154 The command is originated by the devices which want to create one chat room and attract others who have the same  
18155 interest in the topic. It should be unicast to the server which manages the chat rooms.

### 18156 12.3.2.3.6 ChatMessage Command

18157 The *ChatMessage* command is used for chatting, i.e., one node to send a message to other nodes. In the case of peer  
18158 chatting, such as to exchange some private messages, it may be unicast to the chairman first, the chairman may forward  
18159 this command with unicast method to the destination user. The *ChatMessage* command may be sent directly to the  
18160 destination node in peer chatting case if the destination network address and endpoint are known by the sender. *Get*  
18161 *Node Information Request* and *Response* commands shall be used to acquire the necessary network address and  
18162 endpoint information. In the case of normal chatting (a message to be sent to the whole room), all nodes in the same  
18163 chat room are expected to receive the message. The *ChatMessage* command should be multicast to other nodes in the  
18164 chat room.

18165 In peer chatting case, if the command contains illegal parameter such as non-existing U\_ID field, the chairman should  
18166 return a Default Response command with INVALID\_FIELD status.

18167 The *ChatMessage* command shall be formatted as illustrated in Figure 12-40.



18168

**Figure 12-40. Format of the ChatMessage Command**

<b>Octets</b>	2	2	2	Variable	Variable
<b>Data Type</b>	uint16	uint16	uint16	string	string
<b>Field Name</b>	Destination U_ID	Source U_ID	C_ID	Nickname	Message

18169

18170 The Destination U\_ID field indicates the destination node’s U\_ID. The Source U\_ID field indicates the source node’s  
 18171 U\_ID. The C\_ID indicates the ID of the chat room which the sender belongs to. The Nickname field indicates the  
 18172 sender’s Nickname, which shall be in Character string data type.

18173 In the case of peer chatting, the Destination U\_ID field and the Source U\_ID field shall be set to the specific nodes’  
 18174 U\_ID. In the case of normal chatting (sending a message to all the users in the chat room), Destination U\_ID shall be  
 18175 set to 0xffff while Source U\_ID shall be set to the specific source node’s U\_ID.

18176 **12.3.2.3.7 Get Node Information Request Command**

18177 The Get Node Information Request command is used for peer chatting to get the network address and endpoint number  
 18178 of the peer node, so as to use ChatMessage command to send private message to the node. When one wants to send  
 18179 private message to another node in the same chatting session, it shall check whether it has that node’s network address  
 18180 and endpoint number. If not, it shall send this command to the server.

18181 The Get Node Information Request command shall be formatted as illustrated in Figure 12-41.

18182 **Figure 12-41. Format of the Get Node Information Request Command**

<b>Octets</b>	2	2
<b>Data Type</b>	uint16	uint16
<b>Field Name</b>	C_ID	U_ID

18183

18184 The C\_ID field indicates the ID of the chat room which the investigated node belongs to. The U\_ID field indicates the  
 18185 U\_ID of the node to be investigated.

18186 This command should be unicast to the chairman node. A chatting table should be maintained by the chairman. It may  
 18187 be also maintained by other nodes. When a chairman has assigned a U\_ID to a node it shall add related information  
 18188 into the chatting table, and when a node leaves the chatting session it shall remove the record of the leaving device  
 18189 from the table. A node may get the address of another node from the chairman by using the Get Node Information  
 18190 Request command. Once it gets the information, the node may store it for future usage. The detail format of the  
 18191 chatting table is implementer dependent. An example of each item of the table may be illustrated as Figure 12-42. The  
 18192 node number field indicates the number of NodeInformation field. The NodeInformation field is as specified in Figure  
 18193 12-50.

18194 **Figure 12-42. Format of an Item of the Chatting Table**

C_ID	Node number	NodeInformation 1	...	NodeInformation n
------	-------------	-------------------	-----	-------------------

18195 **12.3.2.4 Commands Generated**

18196 The generated commands IDs for the Chatting cluster are listed in Table 12-17.

18197 **Table 12-17. Generated Command IDs for the Chatting Cluster**

Command Identifier Field Value	Description	M/O
0x00	Start Chat Response	O
0x01	Join Chat Response	M
0x02	User Left	M
0x03	User Joined	M
0x04	Search Chat Response	M
0x05	Switch Chairman Request	O
0x06	Switch Chairman Confirm	O
0x07	Switch Chairman Notification	O
0x08	Get Node Information Response	O

18198 **12.3.2.4.1 Start Chat Response Command**18199 The Start Chat Response command is used for server to response to the Start Chat request command. If successful,  
18200 the server shall then form a new chat room and make itself the chairman.

18201 The Start Chat Response command shall be formatted as illustrated in Figure 12-43.

18202 **Figure 12-43. Format of the Start Chat Response Command**

<b>Octets</b>	1	0/2
<b>Data Type</b>	enum8	uint16
<b>Field Name</b>	Status	C_ID

18203

18204 The Status field indicates the status of the previous request. . If it is SUCCESS, the C\_ID field shall exist, or else the  
18205 C\_ID field shall not exist. The C\_ID field indicates the unique identification of the chat room and it is assigned by the  
18206 server. If the server doesn't permit to add a new chat room, i.e., the attribute EnableAddChat being set to FALSE, the  
18207 server shall return this command with FAILURE Status.

18208 This command shall be unicast to the requester.

18209 **12.3.2.4.2 Join Chat Response Command**

18210 The Join Chat Response is used for server to response the Join Chat Request command.

18211 The Join Chat Response shall be formatted as illustrated in Figure 12-44.

18212

**Figure 12-44. Format of the Join Chat Response Command**

<b>Octets</b>	1	2	0/2	Variable	Variable	0/2	Variable
<b>Data Type</b>	enum8	uint16	uint16	string	-	uint16	string
<b>Field Name</b>	Status	C_ID	U_ID 1	Nickname 1	...	U_ID <i>n</i>	Nickname <i>n</i>

18213

18214 The Status field indicates the status of the previous request.. If it is SUCCESS, the list of the U\_ID and Nickname  
 18215 fields shall exist, or else the list shall not exist. The C\_ID field indicates the ID of the chat room which the server  
 18216 manages. It shall be the same as the C\_ID field in the corresponding Join Chat Request command. The list of the U\_ID  
 18217 and Nickname fields indicate other participants in the chat room. Each U\_ID field and the Nickname field respectively  
 18218 indicate the unique ID and the nickname of each user in the chat room.

18219 This command shall be unicast to the requester. After receiving this command, the node should check whether it has  
 18220 received the Add Group command from the chairman. If not, it should wait for that command so as to know which  
 18221 group it belongs to. How long it should wait for the command is specific to the implementation.

18222 **12.3.2.4.3 User Left Command**

18223 The User Left command is used for server to inform other participants that someone has left the chat room.

18224 The User Left shall be formatted as illustrated in Figure 12-45.

18225

**Figure 12-45. Format of the User Left Command**

<b>Octets</b>	2	2	Variable
<b>Data Type</b>	uint16	uint16	string
<b>Field Name</b>	C_ID	U_ID	Nickname

18226

18227 The C\_ID indicates the ID of the chat room which the user left. The U\_ID field indicates the left participant’s unique  
 18228 ID in the chat room. The Nickname field is the nickname of the left participant.

18229 The command shall be multicast to all users in the same chat room.

18230 **12.3.2.4.4 User Joined Command**

18231 The User Joined command is used for server to inform other participants that someone has just joined the chat room.

18232 The User Joined command shall be formatted as illustrated in Figure 12-46.

18233

**Figure 12-46. Format of the User Joined Command**

<b>Octets</b>	2	2	Variable
<b>Data Type</b>	uint16	uint16	string
<b>Field Name</b>	C_ID	U_ID	Nickname

18234

18235 The C\_ID indicates the ID of the chat room which the newcomer joined. The U\_ID field indicates the newcomer's  
18236 unique ID in the chat room. The Nickname field is the nickname of the newcomer.

18237 This command should be multicast to all users in the same chat room. When the newcomer receives the command, it  
18238 shall compare the U\_ID field in the command with U\_ID of itself, if same, it shall ignore the command.

### 18239 12.3.2.4.5 Search Chat Response Command

18240 The Search Chat Response command is used for server to respond to the Search Chat Request command.

18241 The Search Chat Response command shall be formatted as illustrated in Figure 12-47.

18242

**Figure 12-47. Format of the Search Chat Response command**

<b>Octets</b>	1	0/2	Variable	Variable	0/2	Variable
<b>Data Type</b>	map8	uint16	string	...	uint16	string
<b>Field Name</b>	Options	C_ID 1	Name 1	...	C_ID <i>n</i>	Name <i>n</i>

18243

18244 The Options field indicates the options of this command. Bit 0 of the Options field indicates whether the server permits  
18245 other users to add new chat rooms in it. The value 0b0 means permit while 0b1 means not permit. Other bits of this  
18246 field are reserved. The list of the C\_ID and Name fields indicates the information of the available chat rooms. Each  
18247 C\_ID field and Name field indicate respectively the chat room identification and topic of each available chat room.  
18248 It's recommended at least one chat room should be maintained by the server. If no chat room is maintained, the list of  
18249 C\_ID and Name fields shall not exist.

18250 This command should be unicast to the requester. Only the chairman can send out this command, and the list of chat  
18251 room information shall only contain the information of the chat rooms which it manages. The server may also  
18252 broadcast this command to notify other users which chat rooms it manages. After receiving this command, the network  
18253 address and endpoint number should be extracted from the network layer header and APS header, so as to acquiring  
18254 the chairman's network address and endpoint number.

### 18255 12.3.2.4.6 Switch Chairman Request Command

18256 In the case of Ad-Hoc chat, when a chairman wants to leave the chat session, he can use this command to appoint a  
18257 new chairman out of the participating Devices which can continue to manage the chat room.

18258 The Switch Chairman Request command shall be multicast to every device which is in the same chat room. It shall be  
18259 formatted as illustrated in Figure 12-48.

18260

**Figure 12-48. Format of the Switch Chairman Request Command**

<b>Octets</b>	2
<b>Data Type</b>	uint16
<b>Field Name</b>	C_ID

18261 The C\_ID field indicates the ID of the chat room where the chairman is requested to be changed.

18262 **12.3.2.4.7 Switch Chairman Confirm Command**

18263 The Switch Chairman Confirm command is used by the old chairman to inform the node which the chairman has  
 18264 selected to be the new chairman.

18265 The Switch Chairman Confirm command shall be formatted as illustrated in Figure 12-49.

18266 **Figure 12-49. Format of the Switch Chairman Confirm Command**

<b>Octets</b>	2	Variable	Variable	Variable
<b>Data Type</b>	uint16	-	...	-
<b>Field Name</b>	C_ID	NodeInformation 1	...	NodeInformation <i>n</i>

18267  
 18268 The C\_ID field indicates the ID of the chat room which the chairman manages. The NodeInformation field is formatted  
 18269 as illustrated in Figure 12-50. Each *NodeInformation* field contains information about a node participating in the chat  
 18270 session. This field shall contain the following sub-fields, the U\_ID sub-field, *Address* sub-field, *Endpoint* sub-field  
 18271 and the *Nickname* sub-field. The U\_ID sub-field, *Address* sub-field, *Endpoint* sub-field and *Nickname* sub-field  
 18272 indicate the node’s unique ID, network address, endpoint number and nickname respectively. This command shall be  
 18273 unicast to the new chairman.

18274 **Figure 12-50. Format of the NodeInformation Field**

<b>Octets</b>	2	2	1	Variable
<b>Data Type</b>	uint16	data16	uint8	string
<b>Sub-field Name</b>	U_ID	Address	Endpoint	Nickname

18275 **12.3.2.4.8 Switch Chairman Notification Command**

18276 The Switch Chairman Notification command is used by the old chairman to inform other participants in the chat room  
 18277 about the change in the chairman. The Switch Chairman Confirm command shall be formatted as illustrated in Figure  
 18278 12-51.

18279

**Figure 12-51. Format of the Switch Chairman Notification Command**

<b>Octets</b>	2	2	2	1
<b>Data Type</b>	uint16	uint16	data16	uint8
<b>Field Name</b>	C_ID	U_ID	Address	Endpoint

18280

18281 The C\_ID field is the ID of the chat room which the chairman manages. The U\_ID field is the unique ID of the node  
18282 which is the new chairman of the chat room. The *Address* field and the *Endpoint* field are the network address and the  
18283 endpoint number of the new chairman respectively. The command should be multicast to other nodes in the same chat  
18284 room.

### 18285 12.3.2.4.9 Get Node Information Response Command

18286 The Get Node Information Response command is used by the server to give a response to the Get Node Information  
18287 Request command, so that the requesting node can obtain the desired information including network address and  
18288 endpoint number of a specific node. If successful, the server shall provide the node information in the response  
18289 command.

18290 The Get Node Information Response command shall be formatted as illustrated in Figure 12-52.

18291

**Figure 12-52. Format of the Get Node Information Response Command**

<b>Octets</b>	1	2	2	0/2	0/1	Variable
<b>Data Type</b>	enum8	uint16	uint16	data16	uint8	string
<b>Field Name</b>	Status	C_ID	U_ID	Address	Endpoint	Nickname

18292

18293 The *Status* indicates the status of the previous request. If it is SUCCESS, the *Address* field, the *Endpoint* field and the  
18294 *Nickname* field shall exist, or else those fields shall not exist. The C\_ID field and the U\_ID field shall be the same as  
18295 the corresponding *Get Node Information Request* command. The C\_ID field is the ID of the chat room which the  
18296 investigated node belongs to. The U\_ID field is the unique ID of the investigated node. The *Address* field, the *Endpoint*  
18297 field and the *Nickname* field indicate the network address, the endpoint number and the nickname of the investigated  
18298 node respectively. The command shall be unicast to requester. After receiving this command, the node may store the  
18299 information of the investigated node for future usage.

## 18300 12.3.3 Client

### 18301 12.3.3.1 Commands Received

18302 The client receives the cluster specific commands detailed in 12.3.2.4 as required by application profiles.

### 18303 12.3.3.2 Commands Generated

18304 The client generates the cluster specific commands detailed in 12.3.2.3 as required by application profiles.

18305 **12.4 Voice Over ZigBee**

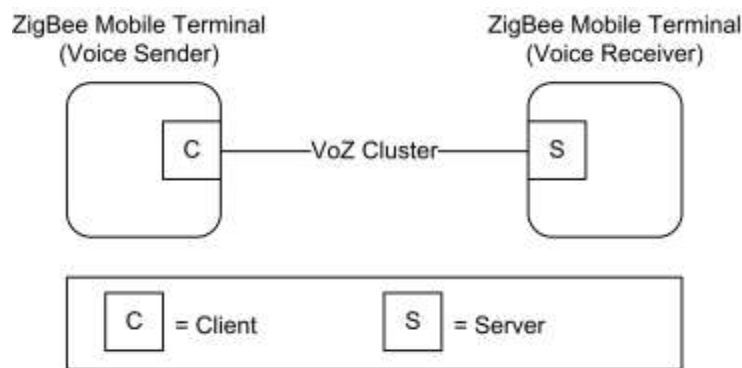
18306 **12.4.1 Scope and Purpose**

18307 This section specifies a single cluster, the VoZ cluster, which provides commands and attributes for voice receiving and transmitting among ZigBee devices. This cluster is to provide a standardized interface for the devices to receive/transmit voice data packets.  
 18308  
 18309

18310 This section should be used in conjunction with Chapter 2, Foundation, which gives an overview of the library and specifies the frame formats and general commands used therein.  
 18311

18312 The cluster specified in this document is typically used for telecom applications, but may be used in any other application domains. This cluster may use Partition cluster.  
 18313

18314 **Figure 12-53. Typical Usage of the VoZ Cluster**



*Note: Device names are examples for illustration purposes only*

18315

18316 **12.4.2 Overview**

18317 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification, etc.  
 18318

18319 This cluster provides attributes and commands for devices to receive/transmit their voice data. One of the devices plays a role of a receiver and the other does that of a sender. For example, a receiver receives voice data from the other MT (voice sender).  
 18320  
 18321

18322 An important thing to notice for this VoZ cluster is that there are two different types of service for this cluster. One of them is voice transmission between humans (human-to-human). The other type of the service is voice transmission from human to device (human-to-device voice data transmission, i.e., voice command) in order to send a voice 'command' to a device. Therefore, the meaning of 'voice' delivery includes not only human voice delivery (human-to-human), but also voice delivery for device control (human-to-device, i.e., voice command). These two types of service will be referenced whenever necessary.  
 18323  
 18324  
 18325  
 18326  
 18327

18328 **12.4.2.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

18329 **12.4.2.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	VOZ	Type 1 (client to server)

18330 **12.4.2.3 Cluster Identifiers**

Identifier	Name
0x0904	Voice Over ZigBee

18331 **12.4.3 Server**

18332 The server stores the data to be shared. It may response to the request from other devices and transmit the data to them,  
18333 or it may actively request other devices to transmit the data to them.

18334 **12.4.3.1 Dependencies**

18335 None

18336 **12.4.3.2 Attributes**

18337 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
18338 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
18339 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
18340 are listed in Table 12-18.

18341 **Table 12-18. VoZ Attribute Sets**

Attribute Set Identifier	Description
0x000	Voice Information
0x001 – 0xffff	Reserved

18342 **12.4.3.2.1 Establishment Information Attribute Set**

18343 The Establishment Information attribute set contains the attributes summarized in Table 12-19.

18344 **Table 12-19. Attributes of the Voice Information Attribute Set**

Id	Name	Type	Range	Access	M/O
0x0000	<i>CodecType</i>	enum8	G.711(PCM) =0x01 G.726(ADPCM)=0x02 CELP =0x03 AMR =0x04	RW	M
0x0001	<i>SamplingFrequency</i>	enum8	SF_8K =0x01 SF_7K =0x02 SF_3_5K =0x03	RW	M



Id	Name	Type	Range	Access	M/O
0x0002	<i>Codecrate</i>	enum8	CR_64K =0x01 CR_40K =0x02 CR_32K =0x03 CR_24K =0x04 CR_16K =0x05 CR_8K =0x06 CR_6_3K =0x07 CR_5_3K =0x08 CR_AMR-NB =0x09 CR_AMR-WB=0x0a	RW	M
0x0003	<i>Establishment Timeout</i>	uint8	0x01-0xff	-	M
0x0004	<i>CodecTypeSub1</i>	enum8	-	RW	O
0x0005	<i>CodecTypeSub2</i>	enum8	-	RW	O
0x0006	<i>CodecTypeSub3</i>	enum8	-	RW	O
0x0007	<i>CompressionType</i>	enum8	ALaw =0x01 uLaw =0x02	-	O
0x0008	<i>CompressionRate</i>	enum8	-	-	O
0x0009	<i>OptionFlags</i>	map8	0x00-0xff	RW	O
0x000a	<i>Threshold</i>	uint8	0x00-0xff	RW	O

18345 **12.4.3.2.1.1 CodecType Attribute**

18346 The *CodecType* attribute specifies the enumeration of the codec type. G.711 Codec is PCM (pulse code modulation)  
 18347 method by the ITU-T. G.726 Codec is ADPCM (adaptive differential PCM) method which has involved G.721 and  
 18348 G.723 ITU-T codec. CELP (code excited linear prediction) is voice codec of CDMA-based digital mobile  
 18349 communication system. AMR (adaptive multirate codec) is used to 3GP European mobile equipment.

18350 **12.4.3.2.1.2 SamplingFrequency Attribute**

18351 The *SamplingFrequency* attribute specifies the enumeration of the sampling frequency (Hz).  
 18352 PCM, ADPCM, CELP: 8KHz, AMR: 3.5KHz, 7KHz

18353 **12.4.3.2.1.3 CodecRate Attribute**

18354 The *CodecRate* attribute specifies the enumeration of the codec rate (Kbps).  
 18355 Various codec rates available.  
 18356 PCM: 64Kbps, ADPCM: 40/32/24/16Kbps, CELP: 5.3/8Kbps, AMR: 5 ~ 12Kbps

18357 **12.4.3.2.1.4 EstablishmentTimeout Attribute**

18358 The *EstablishmentTimeout* attribute sets timeout value to 1/10 sec in order to disconnect an establishment between  
 18359 devices when there is no response after the establishment.

18360 **12.4.3.2.1.5 CodecTypeSub1, CodecTypeSub2, CodecTypeSub3 Attribute**

18361 *CodecTypeSub1*, *CodecTypeSub2*, and *CodecTypeSub3* attributes are used for additionally supportable Codecs other  
 18362 than the system default one. It has the same range value as that of *CodecType* attribute.

18363 **12.4.3.2.1.6 CompressionType Attribute**18364 The *CompressionType* attribute specifies the enumeration of the compression type

18365 ALaw: the compression technology for transmission data to minimize the quantification error in PCM, (Europe)

18366 uLaw: the compression technology for transmission data to minimize the quantification error in PCM, (US, Japan)

18367 **12.4.3.2.1.7 CompressionRate Attribute**18368 The *CodecRate* attribute specifies the enumeration of compression rate.

18369 Compression rate is defined based on compression type.

18370 **12.4.3.2.1.8 OptionFlags Attribute**18371 The *OptionFlags* attribute indicates the optional function. It shall be formatted as illustrated in Figure 12-54.18372 **Figure 12-54. Format of the OptionFlags Attribute**

Bits: b0	b1	b2	b3-b7
Occupancy	PLC	VAD	Reserved

18373 The Occupancy field specifies whether the occupancy sensor is active or not. If the Occupancy field is set to one, it  
18374 indicates the occupancy sensor is active. If the Occupancy field is set to zero, it indicates the occupancy sensor is  
18375 inactive.18376 PLC (Packet Loss Concealment): enabled in logic level high in order to correct voice data when there is loss for the  
18377 data18378 VAD (Voice Activity Detection): enabled in logic level high in order to distinguish mute voice data from non-mute  
18379 voice data18380 **12.4.3.2.1.9 Threshold Attribute**18381 The *Threshold* attribute specifies the value for voice loudness in voice transmission.18382 **12.4.3.3 Commands Received**

18383 The received command IDs for the VoZ cluster are listed in Table 12-20.

18384 Before proceeding, please refer to the section 12.4.2 of overview, and especially, to the service type that there are two  
18385 types of service in this cluster. The commands in this section are developed and used not only for human-to-human  
18386 voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.18387 **Table 12-20. Command IDs for the VoZ Cluster**

Command Identifier Field Value	Description	M/O
0x00	Establishment Request	M
0x01	Voice Transmission	M
0x02	Voice Transmission Completion	O
0x03	Control Response	O

18388 **12.4.3.3.1 Establishment Request Command**

18389 The Establishment Request command is used for a device to request for a connection of the voice information from  
 18390 another device. It shall be originated by the voice transmission source device and sent to the transmission destination  
 18391 device.

18392 The establishment request command shall be formatted as illustrated in Figure 12-55.

18393 For Codec Type, Sampling Frequency, Codec Rate and etc., please refer to Table 12-19. For Service Type, please  
 18394 refer to section 12.4.2; the Service Type equal to 0x00 indicates human-human service and 0x01 indicates human-  
 18395 device service. Most commands in this section are developed and used for human-to-device communication.

18396 **Figure 12-55. Format of the Establishment Request Command**

Octets	Data Type	Field Name
1	map8	Flag
1	enum8	Codec Type
1	enum8	Samp. Freq.
1	enum8	Codec Rate
1	enum8	Service Type
1/0	enum8	Codec TypeS1
1/0	enum8	Codec TypeS2
1/0	enum8	Codec TypeS3
1/0	enum8	Comp. Type
1/0	enum8	Comp. Rate

18397  
 18398 The Flag field value of Figure 12-55 is set according to the bit values of Figure 12-56 when a VoZ device has an  
 18399 optional attribute value such as CodecTypeSub1.

18400 **Figure 12-56. Format of the Flag**

Bits: b0	b1	b2	b3	b4-b7
CodecTypeSub1	CodecTypeSub2	CodecTypeSub3	Compression	Reserved

18401 **12.4.3.3.2 Voice Transmission Command**

18402 The Voice Transmission command is used for a device to transmit the voice data to other devices. It shall be originated  
 18403 by the voice transmission source device and sent to the voice transmission destination device. If required, Partition  
 18404 Cluster should be used for this command.

18405 In case of transmitting multiple voice data, the Sequence Number in the ZCL Header should be sequentially increased  
 18406 in order to detect the loss of data and reassemble them.

18407

Figure 12-57. Format of the Voice Transmission Command

<b>Octets</b>	Variable
<b>Data Type</b>	-
<b>Field Name</b>	Voice Data

18408 **12.4.3.3.3 Voice Transmission Completion**

18409 The Voice Transmission Completion command is sent to the destination device when needed, after the source device  
18410 transmits all voice data which should be transmitted.

18411 The voice transmission command completion shall be formatted as illustrated in Figure 12-58.

18412

Figure 12-58. Format of the Voice Transmission Completion Command

<b>Octets</b>	Variable
<b>Data Type</b>	-
<b>Field Name</b>	ZCL Header

18413 **12.4.3.3.4 Control Response Command**

18414 The Control Response command is used to respond with the success or failure of the control, when a device receives  
18415 the Control command.

18416 The voice control response command shall be formatted as illustrated in Figure 12-59.

18417

Figure 12-59. Format of the Control Response Command

<b>Octets</b>	1
<b>Data Type</b>	enum8
<b>Field Name</b>	ACK=0x01 NAK=0x00

18418 **12.4.3.4 Commands Generated**

18419 The generated command IDs for the VoZ cluster are listed in Table 12-21.

18420 Before proceeding, please refer to the section 12.4.2 of overview, and especially, to the service type that there are two  
18421 types of service in this cluster. The commands in this section are developed and used not only for human-to-human  
18422 voice delivery, but also for human-to-device voice delivery in order to send a voice command to control the device.

18423

**Table 12-21. Generated Command IDs for the VoZ Cluster**

Command Identifier Field Value	Description	M/O
0x00	Establishment Response	M
0x01	Voice Transmission Response	M
0x02	Control	O

18424 **12.4.3.4.1 Voice Transmission Response Command**

18425 The Voice Transmission Response command is to notify the sender of NACK. It shall be originated by the voice  
 18426 transmission destination device and sent to the voice transmission source device.

18427 The voice transmission response command shall be formatted as illustrated in Figure 12-60.

18428 **Figure 12-60. Format of the Voice Transmission Response Command**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	enum8
<b>Field Name</b>	Sequence Number of ZCL Header	Error Flag

18429

18430 If there is an error in processing the received voice data, the receiving device should respond with the Voice  
 18431 Transmission Response command with the sequence number of ZCL Header and the Error Flag set to an error reason  
 18432 according to Table 12-22.

18433

**Table 12-22. The Error Flag of Voice Transmission Response**

Error Flag Identifier Field Value	Description
0x00	Failure to decode voice data
0x01	Wrong order of voice data

18434 **12.4.3.4.2 Establishment Response Command**

18435 The Establishment Response command is to notify the device which previously requests for connecting the voice  
18436 information. It shall be originated by the voice transmission destination device and sent to the voice transmission  
18437 source device.

18438 The Establishment Response command shall be formatted as illustrated in Figure 12-61.

18439 **Figure 12-61. Format of the Establishment Response Command**

<b>Octets</b>	1	1/0
<b>Data Type</b>	enum8	enum8
<b>Field Name</b>	ACK=0x01 NAK=0x00	CodecType

18440

18441 When a receiving device receives the Establishment Request command with *CodecType* which is not supported, it  
18442 responds with the Establishment Response command with NAK and *CodecType* supported by the device.

18443 If the requested *CodecType* exists among *CodecTypeSub1*, *CodecTypeSub2*, and *CodecTypeSub3*, the *CodecType* field  
18444 is set to the value.

18445 If the device receives the Establishment Response command with NAK and *CodecType*, it first checks whether it  
18446 supports the *CodecType* in the received command. If it supports, the device transmits the Establishment Request  
18447 command with its *CodecType* again.

18448 **12.4.3.4.3 Control Command**

18449 The Control command is to control the voice transmission source device. It shall be originated by the voice  
18450 transmission destination device and sent to the voice transmission source device.

18451 For example, this command is used for such as walkie-talkie communication or radio listening.

18452 The voice control command shall be formatted as illustrated in Figure 12-62.

18453

**Figure 12-62. Format of the Control Command**

<b>Octets</b>	1
<b>Data Type</b>	enum8
<b>Field Name</b>	Control Type

18454

18455 The Control Type field indicates the control options, including the play operation (0x01), the stop operation (0x02),  
18456 and the disconnection operation (0x03). The play operation is to request for starting voice data transmission. The stop  
18457 operation is to request for stopping voice data transmission. The disconnection operation is to terminate the connection  
18458 between the voice transmission source/destination devices.

## 18459 **12.4.4 Client**

### 18460 **12.4.4.1 Command Received**

18461 The client receives the cluster specific commands detailed in 12.4.3.4 as required by application profiles

### 18462 **12.4.4.2 Command Generated**

18463 The client generates the cluster specific commands detailed in 12.4.3.3 as required by application profiles.

18464





# CHAPTER 13 COMMISSIONING

18465  
 18466 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 18467 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 18468 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 18469 using [*Rn*] notation.

## 13.1 General Description

### 13.1.1 Introduction

This chapter contains commissioning methods for devices that can be used in any application domain.

### 13.1.2 Cluster List

This section lists the clusters specified in this document. The clusters defined in this document are listed in Table 13-1.

Table 13-1. Clusters for Commissioning

ID	Cluster Name	Description
0x0015	Commissioning	The commands and attributes for commissioning a device onto the network
0x1000	Touchlink	The commands and attributes for Touchlink commissioning a device

## 13.2 Commissioning

### 13.2.1 Overview

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

This cluster provides attributes and commands pertaining to the commissioning and management of devices operating in a network.

This cluster will typically be supported using a “Commissioning Tool.” But, depending on the application and installation scenario, this tool may take many forms. For purposes of this document, any device that implements the client side of this cluster may be considered a commissioning tool.

As with all clusters defined in the Cluster Library, an application may have as many instances of this cluster as needed and may place them on any addressable endpoint.

This cluster is exclusively used for commissioning the ZigBee stack and defining device behavior with respect to the ZigBee network. It does not apply to applications operating on those devices.

18489 **13.2.1.1 Security and Authorization**

18490 The attributes and commands covered in this cluster specification are critical to the operation of a ZigBee device. An  
 18491 application entity that receives a request to access the attributes of this cluster or to execute one of the commands  
 18492 described in sub-clause 13.2.2.3 shall determine whether the originator is authorized to make that request and whether  
 18493 the security processing applied to the received frame was appropriate. The method or methods whereby this is  
 18494 accomplished are out of the scope of this document but it is strongly recommended that Entity Authentication, as  
 18495 described in [B1], be used. This, and any other methods used to authorize commissioning tools and other devices  
 18496 acting as a client for this cluster, shall be detailed in any Application Profile documents that use it.

18497 Similarly, it is strongly recommended that the cluster specified here be deployed only on a single device endpoint or  
 18498 that, at very least, all deployments of this cluster be managed by a single application object with a unitary set of  
 18499 security requirements etc.

18500 **13.2.1.2 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

18501 **13.2.1.3 Classification**

Hierarchy	Role	PICS Code
Base	Utility	CS

18502 **13.2.1.4 Cluster Identifiers**

Identifier	Name
0x0015	Commissioning

18503 **13.2.2 Server**

18504 The attributes accessible on the server side of this cluster are typically attributes of the ZigBee stack, which are either  
 18505 described in the layer Information Base for some stack layer, or are ZDO configuration attributes. The function of the  
 18506 server is to provide read/write access to these attributes and to manage changes of certain critical attributes in a way  
 18507 that prevents the device from getting into an inconsistent and unrecoverable state.

18508 Thus, for example, the application entity that receives and processes commands to set attributes in the Startup  
 18509 Parameters attribute set shall check whether the *StartupControl* attribute has been set to a value that is inconsistent  
 18510 with the value of the *ExtendedPanID* attribute (see Table 13-2). If such a condition arises, e.g., a request is made to  
 18511 set the *StartupControl* attribute to 0x02, indicating network rejoin, and simultaneously to clear the *ExtendedPanID*  
 18512 attribute indicating an unspecified network, then an error (INCONSISTENT\_STARTUP\_STATE) shall be reported.

18513 **13.2.2.1 Dependencies**

18514 None

18515 **13.2.2.2 Attributes**

18516 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 18517 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 18518 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 18519 are listed in Table 13-1.

18520 **Table 13-1. Commissioning Attribute Sets**

Attribute Set Identifier	Description
0x000, 0x001	Startup Parameters
0x002	Join Parameters
0x003	End Device Parameters
0x004	Concentrator Parameters

18521  
 18522 For each of these sets, each attribute is mandatory unless specifically specified as optional in the relevant sub-clause  
 18523 defining it. Similarly, any default values are specified in these sub-clauses.

18524 **13.2.2.2.1 Startup Parameters Attribute Set**

18525 The Startup Parameters attribute set contains the attributes summarized in Table 13-2.

18526 These are application attributes and, as such, are sent, received and managed by application entities. However, except  
 18527 where otherwise noted, each of them corresponds to, and is intended to provide a value for a particular stack attribute  
 18528 that controls the startup behavior of the stack. The ZigBee specification describes a schematic startup procedure (see  
 18529 [B1]), which governs the order and manner in which these stack attributes must be used in order to gain access to a  
 18530 network or form a new network. This procedure should run when a device starts up, but may also run without an actual  
 18531 restart as part of the ongoing operation of the device.

18532 The Restart Device command (see 13.2.2.3.1) provides a means whereby a set of Startup Parameters - the “current”  
 18533 Startup Parameters attribute set - stored at the application layer, can be installed in the stack and put into force by  
 18534 executing the startup procedure described above and in the specification. A change to one of the attributes contained  
 18535 in this set, e.g., the *ShortAddress* attribute, does not immediately result in a change to the underlying stack attribute.  
 18536 The attribute set will be installed on receipt of a Restart Device command.

18537 Note that the attributes in this set are mutually interdependent and must be taken as a whole. One consequence of this  
 18538 is that, while there are no explicit requirements with regard to storage class for these attributes, implementers must  
 18539 carefully consider whether to make a particular attribute non-volatile or static in order to prevent inconsistencies in  
 18540 the attribute set after an unintentional processor restart. Another consequence is that, wherever possible, startup  
 18541 attributes should be written atomically using a single write attributes command frame.

18542 **Table 13-2. Attributes of the Startup Parameters Attribute Set**

Id	Name	Type	Range	Def	Acc	MO
0x0000	ShortAddress	uint16	0x0000 – 0xffff7	-	RW	M
0x0001	ExtendedPANId	EUI64	0x0000000000000000 - 0xfffffffffffe	0xffffffffffffff	RW	M

Id	Name	Type	Range	Def	Acc	MO
0x0002	PANId	uint16	0x0000 - 0xffff	-	RW	M
0x0003	Channelmask	map32	Any valid IEEE 802.15.4 channel mask (see [E1]).	-	RW	M
0x0004	ProtocolVersion	uint8	0x02	-	RW	M
0x0005	StackProfile	uint8	0x01 - 0x02	-	RW	M
0x0006	StartupControl	enum8	0x00 - 0x03	-	RW	M
0x0010	TrustCenterAddress	EUI64	Any valid IEEE Address	<i>all zeros</i>	RW	M
0x0011	TrustCenterMasterKey	key128	Any 128-bit value	<i>all zeros</i>	RW	O
0x0012	NetworkKey	key128	Any 128-bit value	<i>all zeros</i>	RW	M
0x0013	UseInsecureJoin	bool	FALSE/TRUE	TRUE	RW	M
0x0014	PreconfiguredLinkKey	key128	Any 128-bit value	<i>all zeros</i>	RW	M
0x0015	NetworkKeySeqNum	uint8	0x00 - 0xff	0x00	RW	M
0x0016	NetworkKeyType	enum8	Any valid key type value	-	RW	M
0x0017	NetworkManagerAddress	uint16	Any valid network address	0x000	RW	M

18543

18544 Except where specifically noted, an implementer of this cluster shall provide read access to all attributes of the Startup  
18545 Parameters attribute set. However, if an attempt is made to read an attribute that may not be read, a WRITE\_ONLY  
18546 status value shall be returned.

18547 Even in cases where the commissioning cluster is a mandatory part of a given application profile, an implementer is  
18548 not required to provide write access for all attributes. If write access is not provided, it is assumed that the implementer  
18549 has some other preferred, generally out-of-band, method for setting the value of the underlying stack attribute, and  
18550 that the value returned on read reflects the actual value in use. If an attempt is made to write to such an attribute, a  
18551 DEFINED\_OUT\_OF\_BAND status value shall be returned

#### 18552 **13.2.2.2.1.1 ShortAddress Attribute**

18553 The *ShortAddress* attribute contains the intended 16-bit network address of the device. This attribute corresponds to  
18554 the *nwkShortAddress* attribute of the NIB (see [B1]).

18555 The default value is the value stored in the *nwkShortAddress* attribute of the NIB. When this attribute is not set as part  
18556 of the Restart Device Request command, this default value ensures that the previous short address is preserved. This  
18557 makes it possible for a device to preserve its short address after being commissioned.

18558 Stack profile compatibility for this attribute is described in Table 13-3.

18559

**Table 13-3. Stack Profile Compatibility for the *ShortAddress* Attribute**

StackProfile Value	Supported	Comment
0x01	No	Under the ZigBee stack profile a ZigBee router or device shall obtain a network address from its parent at network formation time.
0x02	Yes	Under the ZigBee PRO stack profile and stochastic addressing a device may, under certain circumstances, generate its own network address and keep it through the joining process (see [B1]). In this case, it may make sense for that address to be provided by a tool if, for example, this will reduce the likelihood of address conflicts.

18560 **13.2.2.2.1.2 ExtendedPANId Attribute**

18561 The *ExtendedPANId* attribute holds the extended PAN Id of the network of which the device should be a member. See  
 18562 13.2.2.2.1.7 for usage details.

18563 The default value of 0xffffffffffff indicates an unspecified value. In the case where a device is required to join a  
 18564 commissioning network on startup, this attribute may be set, under application control, to the global commissioning  
 18565 EPID (see 13.2.4.1).

18566 Depending in the value of the *StartupControl* attribute, this attribute may correspond to the *nwkExtendedPANID*  
 18567 attribute of the NIB (see [B1]) or the *apsUseExtendedPANID* attribute of the AIB (see [B1]).

18568 **13.2.2.2.1.3 PANId Attribute**

18569 The *PANId* attribute holds the PAN Id of the network of which the device should be a member. This attribute  
 18570 corresponds to the *macPANId* attribute of the MAC PIB (see [E1]).

18571 The default value is macPANID.

18572 Stack profile compatibility for this attribute is described in Table 13-4.

18573 **Table 13-4. Stack Profile Compatibility for the *PANId* Attribute**

StackProfile Value	Comment
0x01	Under the ZigBee stack profile, The ZigBee coordinator shall select an appropriate PANID at network formation time. In this case the value of the PANId attribute may be used. A ZigBee router or ZigBee end device shall obtain a PANID from its parent at network join time. In this case, the value of the PANId attribute shall be ignored.
0x02	Under the ZigBee PRO stack profile a ZigBee router or end device that has the <i>StartupControl</i> attribute equal to 0x00, must have the PANId attribute set to the correct value since it has no other way of obtaining it.

18574 **13.2.2.2.1.4 ChannelMask Attribute**

18575 The *ChannelMask* attribute is an IEEE802.15.4 channel mask, see [E1], containing the set of channels the device  
18576 should scan as part of the network join or formation procedures. This attribute corresponds to the *apsChannelMask*  
18577 attribute of the AIB (see [B1]).

18578 The default value is the value of *apsChannelMask*.

#### 18579 **13.2.2.2.1.5 ProtocolVersion Attribute**

18580 The *ProtocolVersion* attribute is used to select the current protocol version for a device that supports multiple versions  
18581 of the ZigBee specification.

18582 This attribute is optional. A device may support a single protocol version or multiple protocol versions at the option  
18583 of the implementer.

18584 Currently only one value, 0x02 denoting ZigBee 2006 and later, is supported. The default value shall be the protocol  
18585 version supported by the application if only one protocol version is supported. Should more than one protocol version  
18586 be supported, the default value may be any of the protocol versions supported.

18587 The *ProtocolVersion* attribute corresponds to a NWK layer constant, *nwkcProtocolVersion*, which is defined as a  
18588 constant because most implementations will support only a single ZigBee protocol version. In this case, the attribute  
18589 will be read-only. However, there is nothing to prevent a device with sufficient resources from supporting more than  
18590 one ZigBee protocol version under control of the commissioning cluster.

#### 18591 **13.2.2.2.1.6 StackProfile Attribute**

18592 The *StackProfile* attribute is used to select the stack profile for the device.

18593 This attribute is optional. A device may only support one stack profile.

18594 Supported values include:

18595       0x01: ZigBee Stack profile

18596       0x02: ZigBee PRO Stack Profile

18597 This attribute corresponds to the *nwkcStackProfile* attribute of the NIB (see [B1]). The default value shall be the stack  
18598 profile supported by the application if only one stack profile is supported. Should more than one stack profile be  
18599 supported, the default value may be any of the stack profiles supported.

#### 18600 **13.2.2.2.1.7 StartupControl Attribute**

18601 The *StartupControl* attribute is an enumerated type that determines how certain other parameters are to be used. Values  
18602 for this attribute and interaction with other attributes are shown in Table 13-5. If an attribute appears in the “required  
18603 attributes” column this indicates that this attribute must be set to a value that is valid for the intended operational  
18604 network in order for this *StartupControl* attribute value to be used. Note that in some cases the default value may be  
18605 sufficient.

18606 If an attribute appears in the “optional attributes” column it means that the attribute value will affect startup or  
18607 operation under the given attribute set but that any value, including the default, is a valid value. If an attribute appears  
18608 in the “ignored attributes” column it means that the value of this attribute has no effect on device startup when the  
18609 *StartupControl* attribute value in the “value” column is in force.

18610

**Table 13-5. StartupControl Attribute Usage**

Value	Description	Required Attributes	Optional Attributes	Ignored Attributes
0x00	Indicates that the device should consider itself part of the network indicated by the <i>ExtendedPANId</i> attribute. In this case it will not perform any explicit join or rejoin operation.	ShortAddress, ExtendedPANId, PANId, TrustCenterAddress, NetworkKey, NetworkKeySeqNum, NetworkKeyType	ChannelMask, UseInsecureJoin, NetworkManagerAddress, TrustCenterMasterKey (required for Stack Profile 2, optional for Stack Profile 1), PreconfiguredLinkKey	-
0x01	Indicates that the device should form a network with extended PAN ID given by the <i>ExtendedPANId</i> attribute.  The AIB attribute <i>apsDesignatedCoordinator</i> (see [B1]) shall be set to TRUE in this case.	ExtendedPANId	PANId, ChannelMask, NetworkManagerAddress, NetworkKey, NetworkKeyType, TrustCenterAddress	ShortAddress, UseInsecureJoin NetworkKeySeqNum, TrustCenterMasterKey, PreconfiguredLinkKey
0x02	Indicates that the device should rejoin the network with extended PAN ID given by the <i>ExtendedPANId</i> attribute.  The AIB attribute <i>apsDesignatedCoordinator</i> (see [B1]) shall be set to FALSE in this case.	ExtendedPANId	ShortAddress, ChannelMask, UseInsecureJoin, NetworkKey, NetworkKeySeqNum, NetworkKeyType TrustCenterAddress, TrustCenterMasterKey, NetworkManagerAddress, PreconfiguredLinkKey	PANId
0x03	Indicates that the device should start “from scratch” and join the network using (unsecured) MAC association.  The AIB attribute <i>apsDesignatedCoordinator</i> (see [B1]) shall be set to FALSE in this case.	-	ExtendedPANId, ChannelMask, PreconfiguredLinkKey	ShortAddress, UseInsecureJoin, PANId, TrustCenterAddress, NetworkKey, NetworkKeySeqNum, NetworkKeyType, NetworkManagerAddress, TrustCenterMasterKey

18611

18612 Note that these values control the execution of the device startup procedure as specified in [B1], sub-clause 2.5.5.5.6.2.  
 18613 See this sub-clause for a detailed description of the operation of this procedure.

18614 The default value of the *StartupControl* attribute for an un-commissioned device is 0x03.

18615 Stack profile compatibility for this attribute is shown in Table 13-6.

18616 **Table 13-6. Stack Profile Compatibility for the *StartupControl* Attribute**

StackProfile Value	StartupControl Value		Comment
	Mandatory	Optional	
0x01	0x01 0x03	0x02	ZigBee networks use tree-structured address assignment and must form, at startup, from the ZigBee coordinator. The “mode” implied by <i>StartupControl</i> = 0 in which a device is essentially preconfigured to run on a network without having to explicitly join in order to get an address or PAN Id is not supported.
0x02	0x01 0x03	0x00 0x02	<i>StartupControl</i> = 0 is supported under the ZigBee Pro stack profile.

18617

18618 **Note:** An implementation shall return an error code of `INVALID_VALUE` when a client attempts to write an  
18619 unsupported *StartupControl* value.

#### 18620 **13.2.2.2.1.8 TrustCenterAddress Attribute**

18621 The trust center address to use when performing security operations on the network whose extended PAN ID is given  
18622 by the *ExtendedPANId* attribute is, in turn, given by the *TrustCenterAddress* attribute.

18623 This attribute corresponds to the *apsTrustCenterAddress* attribute of the AIB (see [B1]).

18624 The default value of 0x0000000000000000 indicates unspecified.

#### 18625 **13.2.2.2.1.9 TrustCenterMasterKey Attribute**

18626 This attribute holds the trust center master key to use during key establishment with the TC of the network with the  
18627 extended PAN ID given by the *ExtendedPANId* attribute.

18628 The default value, i.e., a 128-bit value containing all zeros, indicates that the key is unspecified.

18629 This attribute corresponds to the *MasterKey* element of the key-pair set from the *apsDeviceKeyPairSet* attribute of  
18630 the AIB for which the *DeviceAddress* element corresponds to the value of the *TrustCenterAddress* attribute. (see  
18631 [B1]).

#### 18632 **13.2.2.2.1.10 NetworkKey Attribute**

18633 This attribute supplies the NWK key to use when communicating with the network specified by the *ExtendedPANId*  
18634 attribute. The default value, i.e., a 128-bit value containing all zeros, indicates that the key is unspecified.

18635 This attribute corresponds to the active key from the *nwkSecurityMaterialSet* attribute of the NIB (see [B1]).

#### 18636 **13.2.2.2.1.11 UseInsecureJoin Attribute**

18637 This attribute is a Boolean flag that enables the use of unsecured join as a fallback case at startup time. It corresponds  
18638 to the Boolean AIB attribute *apsUseInsecureJoin* (see [B1]). The default value is TRUE.



18639 **13.2.2.2.1.12 PreconfiguredLinkKey Attribute**

18640 The preconfigured link key is the key between the device and the trust center. The default value, i.e., a 128-bit value  
 18641 containing all zeros, indicates that the key is unspecified.

18642 This attribute corresponds to the LinkKey element of the Key-Pair descriptor contained in the *apsDeviceKeyPairSet*  
 18643 attribute of the AIB (see [B1]).

18644 **13.2.2.2.1.13 NetworkKeySeqNum Attribute**

18645 This attribute sets the network key's sequence number. The default value is 0x00.

18646 This attribute corresponds to the value of the *nwkActiveKeySeqNumber* attribute of the NIB (see [B1]).

18647 **13.2.2.2.1.14 NetworkKeyType Attribute**

18648 This attribute sets the network key's type. It corresponds to the value of the KeyType element of the current security  
 18649 material descriptor corresponding to the Trust Center found in the *nwkSecurityMaterialSet* attribute of the NIB (see  
 18650 [B1]).

18651 The default value is 0x01 when the StackProfile is 0x01 and 0x05 when the StackProfile is 0x02.

18652 **13.2.2.2.1.15 NetworkManagerAddress Attribute**

18653 This attribute sets the address of the Network Manager. It corresponds to the value of the *nwkManagerAddr* attribute  
 18654 of the NIB (see [B1]).

18655 The default value is 0x0000 indicating that, by default, the Network Manager is on the ZigBee coordinator.

18656 **13.2.2.2.2 Join Parameters Attribute Set**

18657 The Join Parameters attribute set contains the attributes summarized in Table 13-7.

18658 These attributes control the details of the network joining process. Each of them, as described below, corresponds to  
 18659 a ZDO configuration attribute, the function and use of which is described in the ZigBee specification (see [B1]).

18660 **Table 13-7. Attributes of the Join Parameters Attribute Set**

<b>Id</b>	<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Def</b>	<b>Acc</b>	<b>M/O</b>
0x0020	ScanAttempts	uint8	0x001 – 0xff	0x05	RW	O
0x0021	TimeBetweenScans	uint16	0x0001 - 0xffff	0x64	RW	O
0x0022	RejoinInterval	uint16	0x0001 - <i>MaxRejoinInterval</i>	0x3c	RW	O
0x0023	MaxRejoinInterval	uint16	0x0001 - 0xffff	0x0e10	RW	O

18661  
 18662 As with the attributes in Table 13-2, an implementer of this cluster shall provide read access to all attributes. The  
 18663 implementer may provide write access. If write access is not provided, it is assumed that the implementer has some  
 18664 other preferred method for setting the value of the underlying stack attribute, and that the value returned on read  
 18665 reflects the actual value in use.

18666 **13.2.2.2.2.1 ScanAttempts Attribute**

18667 The *ScanAttempts* attribute determines how many scan attempts to make before selecting the ZigBee Coordinator or  
 18668 Router to join.

18669 This attribute corresponds to the *:Config\_NWK\_Scan\_Attempts* configuration attribute of the ZDO (see [B1]).

18670 The default value for this attribute is 0x05.

#### 18671 **13.2.2.2.2 TimeBetweenScans Attribute**

18672 The *TimeBetweenScans* attribute determines the time between each scan attempt.

18673 This attribute corresponds to the *:Config\_NWK\_Time\_btwn\_Scans* configuration attribute of the ZDO (see [B1]).

18674 The units of this attribute are milliseconds and the default value is 0x64.

#### 18675 **13.2.2.2.3 RejoinInterval Attribute**

18676 The *RejoinInterval* determines the interval between attempts to rejoin the network if an end device finds itself disconnected.

18678 This attribute corresponds to the *:Config\_Rejoin\_Interval* configuration attribute of the ZDO (see [B1]).

18679 The units of this attribute are seconds and the default value is 0x3c.

#### 18680 **13.2.2.2.4 MaxRejoinInterval Attribute**

18681 The *MaxRejoinInterval* attribute imposes an upper bound on the RejoinInterval parameter.

18682 This attribute corresponds to the *:Config\_Max\_Rejoin\_Interval* configuration attribute of the ZDO (see [B1]).

18683 The units of this attribute are seconds and the default value is 0x0e10.

### 18684 **13.2.2.3 End Device Parameters Attribute Set**

18685 The End Device Parameters attribute set contains the attributes summarized in Table 13-8.

18686 **Table 13-8. Attributes of the End Device Parameters Attribute Set**

Id	Name	Type	Range	Def	Acc	M/O
0x0030	IndirectPollRate	uint16	0x0000 – 0xffff	-	RW	O
0x0031	ParentRetryThreshold	uint8	0x00 - 0xff	-	R	O

18687

18688 As with the attributes in Table 13-2 and Table 13-7, an implementer of this cluster shall provide read access to all  
18689 attributes. The implementer may provide write access. If write access is not provided, it is assumed that the  
18690 implementer has some other preferred method for setting the value of the underlying stack attribute, and that the value  
18691 returned on read reflects the actual value in use.

#### 18692 **13.2.2.2.3.1 IndirectPollRate Attribute**

18693 The *IndirectPollRate* attribute determines the rate at which a device, usually an end device, where the  
18694 *macRxOnWhenIdle* attribute of the PIB has a value of FALSE, will poll for messages from its parent.

18695 This attribute corresponds to the *:Config\_NWK\_IndirectPollRate* configuration attribute of the ZDO (see [B1]).

18696 The units for this attribute are milliseconds and the default value, broad limits for which are given in [Z2] and [Z3],  
18697 shall be determined by the relevant application. Values assigned using this cluster should be within the given limits in  
18698 order to promote correct network operation.

#### 18699 **13.2.2.2.3.2 ParentRetryThreshold Attribute**

18700 The *ParentRetryThreshold* attribute determines how many times a ZigBee end device should attempt to contact its  
18701 parent before initiating the rejoin process. ZigBee routers and ZigBee coordinators should return a value of 0xff for  
18702 this attribute on read, and should return an error on any attempt to write it.

18703 This attribute corresponds to the *:Config\_Parent\_Link\_Retry\_Threshold* configuration attribute of the ZDO (see  
 18704 [B1]).

18705 **13.2.2.2.4 Concentrator Parameters Attribute Set**

18706 The Concentrator Parameters attribute set contains the attributes summarized in Table 13-9.

18707 **Table 13-9. Attributes of the Concentrator Parameters Attribute Set**

Idc	Name	Type	Range	Def	Acc	M/O
0x0040	ConcentratorFlag	bool	FALSE/TRUE	FALSE	RW	O
0x0041	ConcentratorRadius	uint8	0x00 - 0xff	0x0f	RW	O
0x0042	ConcentratorDiscoveryTime	uint8	0x00 - 0xff	0x0000	RW	O

18708  
 18709 As with the other attribute sets in this cluster, an implementer shall provide read access to all attributes. The  
 18710 implementer may provide write access. If write access is not provided, it is assumed that the implementer has some  
 18711 other preferred method for setting the value of the underlying stack attribute, and that the value returned on read  
 18712 reflects the actual value in use.

18713 **13.2.2.2.4.1 ConcentratorFlag Attribute**

18714 The *ConcentratorFlag* attribute will configure the device to be a concentrator for the purpose of many-to-one routing.  
 18715 This attribute corresponds to the *nwkIsConcentrator* attribute of the NIB (see [B1]).

18716 The default value for this attribute is FALSE.

18717 **13.2.2.2.4.2 ConcentratorRadius Attribute**

18718 The *ConcentratorRadius* attribute determines the hop count radius for concentrator route discoveries. This attribute  
 18719 corresponds to the *nwkConcentratorRadius* attribute of the NIB (see [B1]).

18720 The default value for this attribute is 0x0f.

18721 **13.2.2.2.4.3 ConcentratorDiscoveryTime Attribute**

18722 Routes to the concentrator are known as inbound routes. These routes are created after the receipt of a command from  
 18723 the concentrator. The *ConcentratorDiscoveryTime* attribute determines the period for triggering such route creation.

18724 This attribute corresponds to the *nwkConcentratorDiscoveryTime* attribute of the NIB (see [B1]).

18725 The units of this attribute are seconds and the default value is 0x0000, which indicates that the discovery time is  
 18726 unknown and must be performed by the application.

18727 **13.2.2.3 Commands Received**

18728 The received command IDs for the commissioning cluster server are listed in Table 13-10. These commands may, in  
 18729 principle, be received as unicasts or as broadcasts, but application developers should be aware that, since these  
 18730 commands require a response, broadcasting them to a large number of devices may not be advisable.

18731

**Table 13-10. Commands Received by the Commissioning Cluster Server**

Command Identifier	Description	M/O
0x00	Restart Device	M
0x01	Save Startup Parameters	O
0x02	Restore Startup Parameters	O
0x03	Reset Startup Parameters	M

18732

18733 In Table 13-10, if the actions associated with an optional command are not implemented, at least the relevant response  
18734 command (see Table 13-12) must be returned with status UNSUP\_CLUSTER\_COMMAND.

### 18735 13.2.2.3.1 Restart Device Command

18736 The Restart Device command is used to optionally install a set of startup parameters in a device and run the startup  
18737 procedure so as to put the new values into effect. The new values may take effect immediately or after an optional  
18738 delay with optional jitter. The server will send a Restart Device Response command back to the client device before  
18739 executing the procedure or starting the countdown timer required to time the delay.

#### 18740 13.2.2.3.1.1 Payload Format

18741 The Restart Device command is formatted as shown in Figure 13-1.

18742 **Figure 13-1. Format of the Restart Device Command Payload**

<b>Octets</b>	1	1	1
<b>Data Type</b>	8-bit bitmap	Unsigned 8-bit integer	Unsigned 8-bit integer
<b>Field Name</b>	Options	Delay	Jitter

18743

18744 **Figure 13-2. Format of the Options Field**

<b>Bits: 0...2</b>	<b>3</b>	<b>4...7</b>
Startup Mode	Immediate	Reserved

18745

18746 The Startup Mode sub-field of the options field is 3 bits in length and shall take one of the nonreserved values from  
18747 Table 13-11.

18748

**Table 13-11. Startup Mode Sub-field Values**

Field value	Description
0b000	Restart the device using, i.e., installing, the current set of startup parameters.
0b001	Restart the device using, and not replacing, the current state of the device, i.e., the current set of stack attributes.

18749

18750 The Immediate sub-field of the options field is 1 bit in length. If this sub-field has a value of 1 then the device is to  
 18751 execute the restart either immediately on receipt of the Restart Device Request frame, if the value of the delay field is  
 18752 0, or immediately after the prescribed delay and jitter has transpired if not. If the immediate sub-field has a value of  
 18753 0, then the device may wait to restart until after the prescribed delay and jitter, if any, have transpired but may also  
 18754 wait for a “convenient” moment, e.g., until pending frames have been transmitted, to actually perform the restart.

18755 The delay field is one octet in length and gives a delay in seconds, in the range [0...255], after which the startup  
 18756 procedure is to be invoked.

18757 The jitter field is one octet in length and specifies a random jitter range. While possible field values fall in the interval  
 18758 [0...255], the actual jitter, in milliseconds, that should be added to the delay, given in seconds, in the delay field should  
 18759 be:

18760  $RAND(<jitter\ field\ contents> * 80)$  ms.

18761 Where  $RAND(X)$  returns a random number in the interval [0...X].

18762 **13.2.2.3.1.2 Effect on Receipt**

18763 On receipt of the Restart Device command, the application checks the current startup attribute set for consistency. If  
 18764 the attribute set is incorrect or inconsistent, processing of the command is terminated and a Restart Device Response  
 18765 command is returned to the sender of the request with a status value of `INCONSISTENT_STARTUP_STATE`.  
 18766 Otherwise, the application sends a Restart Device Response command to the sender of the request with a status value  
 18767 of `SUCCESS`, then leaves the current network, installs the current startup attribute set, if the startup mode sub-field  
 18768 of the options field has a value of 0b00, and runs the restart procedure after the given delay and jitter have transpired.

18769 **13.2.2.3.2 Save Startup Parameters Command**

18770 In addition to the current set of startup parameters, which every device implementing the commissioning cluster must  
 18771 maintain, a device may store and maintain up to 256 sets of startup attributes. The Save Startup Parameters Request  
 18772 command allows for the current attribute set to be stored under a given index. Note that while the startup attribute set  
 18773 index is 8 bits, allowing for as many as 256 attribute sets, the actual number of attribute sets will typically be much  
 18774 smaller.

18775 While storage of additional startup attribute sets is optional, a device that chooses to store additional startup attribute  
 18776 sets must store them in such a way that they are non-volatile.

18777 **13.2.2.3.2.1 Payload Format**

18778 The Save Startup Parameters command is formatted as shown in Figure 13-3.

18779

**Figure 13-3. Format of Save Startup Parameters Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	8-bit bitmap	Unsigned 8-bit integer
<b>Field Name</b>	Options (Reserved)	Index

18780

18781 The Options field is one octet in length and is reserved.

18782 The Index field is one octet in length and gives an index under which the current startup parameter attribute set is to be saved.

18784 **13.2.2.3.2 Effect on Receipt**

18785 On receipt of the Save Startup Parameters command, the application shall check the value of the index field of the  
18786 command payload. If the index field has a value that is equal to an index under which a set of startup parameters has  
18787 already been saved then the current startup parameters attribute set is simply saved in place of the previously saved  
18788 set and a Save Startup Parameters Response command is sent back to the sender of the request with a status value of  
18789 SUCCESS.

18790 If the value of the index field is such that no startup parameters attribute set has been saved under that index then the  
18791 application shall check that there is storage capacity to save another attribute set. If there is capacity then the current  
18792 startup parameters attribute set shall be stored under the index given in the index field such that it may be restored at  
18793 a future time in response to the receipt of a Restore Startup Parameters Request command carrying the same index. A  
18794 Save Startup Parameters Response command with status value of SUCCESS is then sent as described above.

18795 If there is not storage capacity, then a save Startup Parameters Response command is sent back to the sender of the  
18796 request with a status INSUFFICIENT\_SPACE.

18797 **13.2.2.3.3 Restore Startup Parameters Command**

18798 A device that implements the optional Save Startup Parameters command shall also implement the Restore Startup  
18799 Parameters Request command (and vice-versa). This command allows a saved startup parameters attribute set to be  
18800 restored to current status overwriting whatever was there previously.

18801 **13.2.2.3.3.1 Payload Format**

18802 The Restore Startup Parameters command is formatted as shown in Figure 13-4.

18803

**Figure 13-4. Restore Startup Parameters Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	8-bit bitmap	Unsigned 8-bit integer
<b>Field Name</b>	Options (Reserved)	Index

18804

18805 The options field is one octet in length and is reserved.

18806 The index field is one octet in length and gives the index of the saved startup parameter attribute set to be restored to  
18807 current status.

18808 **13.2.2.3.3.2 Effect on Receipt**

18809 On receipt of the Restore Startup Parameters command, the application shall check the value of the index field of the  
 18810 command payload. If the index field has a value that is equal to an index under which a startup parameters attribute  
 18811 set has been saved then that attribute set is copied into the current startup parameters attribute set overwriting whatever  
 18812 was there and a Restore Startup Parameters Response command is sent back to the sender of the request with a status  
 18813 value of SUCCESS. If the value of the index field is such that no startup parameters attribute set has been saved under  
 18814 that index then a Restore Startup Parameters Response command is sent back to the sender of the request with a status  
 18815 value of INVALID\_FIELD.

18816 **13.2.2.3.4 Reset Startup Parameters Command**

18817 This command allows current startup parameters attribute set and one or all of the saved attribute sets to be set to  
 18818 default values. There is also an option for erasing the index under which an attribute set is saved thereby freeing up  
 18819 storage capacity.

18820 **13.2.2.3.4.1 Payload Format**

18821 The Reset Startup Parameters command is formatted as shown in Figure 13-5.

18822 **Figure 13-5. Format of Reset Startup Parameters Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	8-bit bitmap	Unsigned 8-bit integer
<b>Field Name</b>	Options	Index

18823  
 18824 The Options field is formatted as shown in Figure 13-6.

18825 **Figure 13-6. Format of the Options Field**

<b>Bits: 0</b>	<b>1</b>	<b>2</b>	<b>3...7</b>
Reset Current	Reset All	Erase Index	Reserved

18826  
 18827 The Reset Current sub-field of the options field is 1 bit in length. If it has a value of 1 then all attributes in the current  
 18828 startup parameters attribute set shall be reset to their default values. Otherwise the current startup parameters attribute  
 18829 set shall remain unchanged.

18830 The Reset All sub-field of the options field is 1 bit in length. If it has a value of 1 then all attributes of all saved startup  
 18831 parameter attribute sets shall be reset to their default values. Otherwise, all attributes of the saved attribute set with an  
 18832 index given by the value of the index field shall be set to their default values

18833 The Erase Index sub-field of the options field is 1 bit in length. If it has a value of 1 then the index under which a  
 18834 saved attribute set has been saved shall be cleared as well, essentially freeing the storage associated with that index.

18835 The Index field is one octet in length and gives the index of a saved startup parameter attribute set. The value of this  
 18836 field is ignored if either the reset all sub-field or the reset current sub-field of the options field have a value of 1.

18837 **13.2.2.3.4.2 Effect on Receipt**

18838 On receipt of the Reset Startup Parameters Request command the application interprets the options field and index  
18839 field as described in sub-clause 13.2.2.3.4.1 and acts accordingly. The Reset Startup Parameters Response command  
18840 sent back to the sender of the request shall always have a status value of SUCCESS.

## 18841 **13.2.2.4 Commands Generated**

18842 The command IDs for the commands generated by the commissioning cluster server are listed in Table 13-12.

18843 **Table 13-12. Commands Generated by the Commissioning Cluster Server**

Command Identifier	Description	M/O
0x00	Restart Device Response	M
0x01	Save Startup Parameters Response	M
0x02	Restore Startup Parameters Response	M
0x03	Reset Startup Parameters Response	M

18844  
18845 These commands should always be issued as unicasts.

### 18846 **13.2.2.4.1 Payload Format**

18847 All response commands emitted by the server have the same payload format as shown in Figure 13-7.

18848 **Figure 13-7. Format of Reset Startup Parameters Command Payload**

Octets	1
Data Type	8-bit enumeration
Field Name	Status

18849  
18850 Status values are chosen from the set of non-reserved values shown in Chapter 2

### 18851 **13.2.2.4.2 Effect on Receipt**

18852 On receipt of one of the response commands shown in Table 13-12, the client is made aware that the server has  
18853 received the corresponding request and is informed of the status of the request.

## 18854 **13.2.3 Client**

18855 The commissioning cluster client (e.g., implemented on a commissioning tool) manages the attributes described above  
18856 on a remote device and sends the Restart Device command as necessary.

### 18857 **13.2.3.1 Dependencies**

18858 None



### 18859 **13.2.3.2 Attributes**

18860 The client cluster has no attributes.

### 18861 **13.2.3.3 Commands Received**

18862 The client receives the cluster specific commands generated by the server (see 13.2.2.4).

### 18863 **13.2.3.4 Commands Generated**

18864 The client generates the cluster specific commands received by the server, as required by the application. See 13.2.2.3.

## 18865 **13.2.4 Commissioning EUI-64s**

18866 To assist in ensuring that commissioning can be achieved in an interoperable environment while minimizing the  
18867 possibility of interference from existing or future ZigBee and 802.15.4 networks and devices a range of IEEE-defined  
18868 64-bit extended unique identifiers (EUI-64s), as been reserved for use as Extended PAN IDs. The reserved range is as  
18869 follows:

- 18870 • 00-50-C2-77-10-00-00-00 is the global commissioning EPID
- 18871 • 00-50-C2-77-10-00-00-01 to 00-50-C2-77-10-00-FF-FF are EUI-64s reserved for other commissioning use

### 18872 **13.2.4.1 Global Commissioning EPID**

18873 The global commissioning EPID is intended to serve as a single EUI-64 to be used by any ZigBee application for the  
18874 purpose of commissioning. It is recommended that profile and application developers that require interoperability  
18875 between products offered by different OEMs incorporate this global commissioning EPID within their respective  
18876 application profiles as the EPID that devices attempt to join when they are first turned on straight “out-of-the-box.”

18877 This global commissioning EPID provides a guarantee that devices will join a specific network for commissioning  
18878 purposes. As part of commissioning, devices are then provided with a startup attribute set (SAS) that ensures that they  
18879 join a network other than this global commissioning network. These SASs may be provided over-the-air using the  
18880 commissioning cluster or some other out-of-band method.

18881 It is also recommended that this global commissioning EPID be used only for commissioning, and especially not for  
18882 ongoing operational use. Commissioning networks formed using the global commissioning EPID should be temporary  
18883 and such networks should be stopped upon completion of commissioning to minimize the possibility of such networks  
18884 interfering with other attempts at forming commissioning networks.

### 18885 **13.2.4.2 EUI-64s Reserved for Other Uses**

18886 Additional EUI-64s have been reserved for other use by the Alliance. At this point, their intended usage has not been  
18887 specified. These identifiers should not be used without prior agreement with the ZigBee Alliance. It is recommended  
18888 that if a profile or application developer requires the use of these additional EUI-64s, they should contact the Core  
18889 Stack Group (CSG) within the ZigBee Alliance.

## 18890 **13.3 Touchlink Commissioning**

---

18891 The *Touchlink Commissioning* cluster provides commands to support touchlink commissioning. This cluster should  
18892 not be considered part of a sub-device but rather part of the entire device. The touchlink commissioning cluster is  
18893 comprised of two sets of commands – one providing touchlink commissioning functionality and one providing  
18894 commissioning utility functionality.

18895 The touchlink commissioning command set has command identifiers in the range 0x00 – 0x3f and shall be transmitted  
18896 using the inter-PAN transmission service.

18897 The commissioning utility command set has command identifiers in the range 0x40 – 0xff and shall be transmitted  
18898 using the standard unicast transmission service, similar to that used for other ZCL cluster commands. These commands  
18899 enable the exchange of control information between controllers (i.e., devices with a device identifier in the range  
18900 0x0800 – 0x0850).

18901 A controller application endpoint may send an *endpoint information* command frame to another controller application  
18902 endpoint to announce itself. It is then up to the recipient controller application endpoint to decide to take further action  
18903 to get information about the lights that are controlled by the originator. If it decides to do so, it can use the *get group*  
18904 *identifiers request* command frame to get knowledge about the group of lights controlled by the originator. The  
18905 originator responds with a *get group identifiers response* command frame containing the requested information (which  
18906 may have a start index field and a count field equal to 0, indicating no groups are used). Similarly, the recipient device  
18907 can use the *get endpoint list request* command frame to get knowledge about the list of individual lights controlled by  
18908 the originator. The originator responds with a *get endpoint list response* command frame containing the requested  
18909 information (which may have a start index field and a count field equal to 0, indicating no lights are controlled).

18910 **Note:** A typical controller application will likely reside inside battery powered remote controllers on top of a ZigBee  
18911 sleeping end-device. As such, care should be taken as to when to send these commands to ensure the recipient is  
18912 awake. It is recommended that such commands are sent just after touchlink commissioning between two controllers  
18913 when the devices are not yet asleep and still polling for data from their parent.

## 18914 13.3.1 Overview

18915 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
18916 etc.

18917 The *touchlink commissioning* cluster shall have a cluster identifier of 0x1000. Those commands in the touchlink  
18918 commissioning command set shall be sent using the profile identifier, 0xc05e whereas those commands in the  
18919 commissioning utility command set shall sent using the profile identifier, 0x0104.

### 18920 13.3.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added
2	added Profile Interop bit in Scan Request frame, CCB 2115 2105

### 18921 13.3.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	TL

### 18922 13.3.1.3 Cluster Identifiers

Identifier	Name
0x1000	Touchlink Commissioning

18923 **13.3.2 Server**

18924 **13.3.2.1 Attributes**

18925 The server has no attributes.

18926 **13.3.2.2 Commands Received**

18927 When a device implements the *touchlink commissioning* cluster at the ZCL server side, it shall be able to receive the  
18928 commands listed in Table 13-13.

18929

**Table 13-13. Commands Received by the Server Side of the Touchlink Commissioning Cluster**

	Command Identifier Field Value	Description	M/O	Reference
<b>Touchlink</b>	0x00	Scan request	M	13.3.2.2.1
	0x02	Device information request	M	13.3.2.2.2
	0x06	Identify request	M	13.3.2.2.3
	0x07	Reset to factory new request	M	13.3.2.2.4
	0x10	Network start request	M	13.3.2.2.5
	0x12	Network join router request	M	13.3.2.2.6
	0x14	Network join end device request	M	13.3.2.2.7
	0x16	Network update request	M	13.3.2.2.8
	All other values in the range 0x00 – 0x3f	<i>Reserved</i>	-	-
<b>Utility</b>	0x41	Get group identifiers request	O*	13.3.2.2.9
	0x42	Get endpoint list request	O*	13.3.2.2.10
	All other values in the range 0x40 – 0xff	<i>Reserved</i>	-	-

18930 \* These are mandatory for a controller device as defined in the device specification.

18931 **13.3.2.2.1 Scan Request Command Frame**

18932 The *scan request* command frame is used to initiate a scan for other devices in the vicinity of the originator. The  
18933 information contained in this command frame relates to the scan request initiator.

18934 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
18935 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
18936 field shall be set to 0 (no ACK requested) and 0b10 (short network address), respectively, the destination address field  
18937 shall be set to 0xffff (broadcast network address) and the source PAN ID field shall be set to any value in the range  
18938 0x0001 – 0xffff, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the  
18939 delivery mode sub-field of the frame control field shall be set to 0b10 (broadcast). In the ZCL header, the direction  
18940 sub-field of the frame control field shall be set to 0 (client to server) and the command identifier shall be set to 0x00  
18941 (scan request).

18942 The ZCL payload field shall contain the *scan request* command frame itself, formatted as illustrated in Figure 13-8.

18943 **Figure 13-8. Format of the Scan Request Command Frame**

<b>Octets</b>	4	1	1
<b>Data Type</b>	Unsigned 32-bit integer	8-bit bitmap	8-bit bitmap
<b>Field Name</b>	Inter-PAN transaction identifier	ZigBee information	Touchlink information

18944 **13.3.2.2.1.1 Inter-PAN Transaction Identifier Field**

18945 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 18946 This field shall contain a 32-bit non-zero random number and is used to identify the current transaction.

18947 **13.3.2.2.1.2 ZigBee Information Field**

18948 The *ZigBee information* field is 8-bits in length and specifies information related to ZigBee. This field shall be  
 18949 formatted as illustrated in Figure 13-9.

18950 **Figure 13-9. Format of the ZigBee Information Field**

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

18951 The *logical type* subfield is 2-bits in length and specifies the logical type of the device. The value of this subfield shall  
 18952 be set to 0b00 for a coordinator, 0b01 for a router or 0b10 for an end device.

18953 The Rx on when idle subfield is 1 bit in length and specifies the *RxOnWhenIdle* state of the device. The value of this  
 18954 subfield shall be set to 1 to indicate that the receiver is left on when the device is idle or 0 otherwise.

18955 **13.3.2.2.1.3 Touchlink information field**

18956 The *Touchlink information* field is 8-bits in length and specifies touchlink-specific information. This field shall be  
 18957 formatted as illustrated in Figure 13-10.

18958 **Figure 13-10. Format of the Scan Request Touchlink Information Field**

Bits: 0	1	2-3	4	5	6	7
Factory new	Address assignment	Reserved	Link initiator	Undefined (can be 0 or 1)	Reserved	Profile Interop <sup>156</sup>

18959 The *factory new* subfield is 1 bit in length and specifies whether the device is factory new. The value of this subfield  
 18960 shall be set to 1 to indicate the device is factory new or 0 otherwise.

18961 The address assignment subfield is 1 bit in length and specifies whether the device is capable of assigning addresses.  
 18962 The value of this subfield shall be set to 1 to indicate the device is capable of assigning addresses or 0 otherwise.

18963 The link initiator subfield is 1 bit in length and specifies whether the device is capable of initiating a link operation.  
 18964 The value of this subfield shall be set to 1 to indicate the device is capable of initiating a link (i.e., it supports the  
 18965 touchlink commissioning cluster at the client side) or 0 otherwise (i.e., it does not support the touchlink commissioning  
 18966 cluster at the client side).

18967 The Profile Interop subfield is 1 bit in length and specifies which profile the device implements. If the ZLL  
 18968 profile is implemented, this bit shall be set to 0. In all other case (Profile Interop / ZigBee 3.0), this bit shall  
 18969 be set to 1.<sup>157</sup>

18970

18971 **13.3.2.2.2 Device Information Request Command Frame**

18972 The *device information request* command frame is used to request information regarding the sub-devices of a remote  
 18973 device.

<sup>156</sup> CCB 2115  
<sup>157</sup> CCB 2115

18974 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 18975 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 18976 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 18977 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 18978 the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
 18979 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
 18980 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
 18981 command identifier shall be set to 0x02 (device information request).

18982 The ZCL payload field shall contain the *device information request* command frame itself, formatted as illustrated in  
 18983 Figure 13-11.

18984 **Figure 13-11. Format of the Device Information Request Command Frame**

<b>Octets</b>	4	1
<b>Data Type</b>	Unsigned 32-bit integer	Unsigned 8-bit integer
<b>Field Name</b>	Inter-PAN transaction identifier	Start index

#### 18985 **13.3.2.2.1 Inter-PAN Transaction Identifier Field**

18986 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 18987 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
 18988 command frame sent by the initiator.

#### 18989 **13.3.2.2.2 Start Index Field**

18990 The *start index* field is 8-bits in length and specifies the starting index (starting from 0) into the device table from  
 18991 which to get device information.

#### 18992 **13.3.2.2.3 Identify Request Command Frame**

18993 The *identify request* command frame is used to request that the recipient identifies itself in some application specific  
 18994 way to aid with touchlinking.

18995 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 18996 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 18997 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 18998 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 18999 the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
 19000 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
 19001 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
 19002 command identifier shall be set to 0x06 (identify request).

19003 The ZCL payload field shall contain the *identify request* command frame itself, formatted as illustrated in Figure  
 19004 13-12.

19005

**Figure 13-12. Format of the Identify Request Command Frame**

<b>Octets</b>	4	2
<b>Data Type</b>	Unsigned 32-bit integer	Unsigned 16-bit integer
<b>Field Name</b>	Inter-PAN transaction identifier	Identify duration

19006 **13.3.2.2.3.1 Inter-PAN Transaction Identifier Field**

19007 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 19008 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
 19009 command frame sent by the initiator.

19010 **13.3.2.2.3.2 Identify Duration Field**

19011 The *identify duration* field is 16-bits in length and shall specify the length of time the recipient is to remain in identify  
 19012 mode. The value of this field shall be set to one of the values listed in Table 13-14.

19013 **Table 13-14. Values of the Identify Duration Field**

<b>Identify duration Field Value</b>	<b>Description</b>
0x0000	Exit identify mode
0x0001 – 0xffff	Number of seconds to remain in identify mode
0xffff	Remain in identify mode for a default time known by the receiver

19014 Note that if a device is not capable of identifying for the exact time specified in the identify duration field, it shall  
 19015 identify itself for a duration as close as possible to the requested value.

19016 **13.3.2.2.4 Reset to Factory New Request Command Frame**

19017 The *reset to factory new request* command frame is used to request that the recipient resets itself back to its factory  
 19018 new state.

19019 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 19020 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 19021 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 19022 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 19023 the preceding scan request inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
 19024 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
 19025 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
 19026 command identifier shall be set to 0x07 (reset to factory new request).

19027 The ZCL payload field shall contain the reset to factory new request command frame itself and this shall be formatted  
 19028 as illustrated in Figure 13-13.

19029

**Figure 13-13. Format of the Reset to Factory New Request Command Frame**

<b>Octets</b>	4
<b>Data Type</b>	Unsigned 32-bit integer
<b>Field Name</b>	Inter-PAN transaction identifier

19030 **13.3.2.2.4.1 Inter-PAN Transaction Identifier Field**

19031 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 19032 This field shall contain a non-zero 32-bit random number and is used to identify the current reset to factory new  
 19033 request.

19034 **13.3.2.2.5 Network Start Request Command Frame**

19035 The *network start request* command frame is used by a factory new initiator to form a new network with a router.

19036 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 19037 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 19038 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 19039 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 19040 the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
 19041 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
 19042 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
 19043 command identifier shall be set to 0x10 (network start request).

19044 The ZCL payload field shall contain the *network start request* command frame itself, formatted as illustrated in Figure  
 19045 13-14.

19046 **Figure 13-14. Format of the Network Start Request Command Frame**

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

19047 **13.3.2.2.5.1 Inter-PAN Transaction Identifier Field**



19048 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19049 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
19050 command frame sent by the initiator.

#### 19051 **13.3.2.2.5.2 Extended PAN Identifier Field**

19052 The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the new network.  
19053 If this value is equal to zero, the target shall determine the extended PAN identifier for the new network.

#### 19054 **13.3.2.2.5.3 Key Index Field**

19055 The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the  
19056 protection method) to be used in the *encrypted network key* field.

#### 19057 **13.3.2.2.5.4 Encrypted Network Key Field**

19058 The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network,  
19059 encrypted according to the algorithm indicated by the *key index* field.

#### 19060 **13.3.2.2.5.5 Logical Channel Field**

19061 The *logical channel* field is 8-bits in length and shall contain the touchlink channel to be used for the new network. If  
19062 this value is equal to zero, the target shall determine the logical channel for the new network.

#### 19063 **13.3.2.2.5.6 PAN Identifier Field**

19064 The *PAN identifier* field is 16-bits in length and shall contain the identifier of the new PAN. If this value is equal to  
19065 zero, the target shall determine the PAN identifier for the new network.

#### 19066 **13.3.2.2.5.7 Network Address Field**

19067 The *network address* field is 16-bits in length and contains the short network address (in the range 0x0001 – 0xffff7)  
19068 assigned to the recipient.

#### 19069 **13.3.2.2.5.8 Group Identifiers Begin Field**

19070 The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the  
19071 recipient can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.

#### 19072 **13.3.2.2.5.9 Group Identifiers End Field**

19073 The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the  
19074 recipient can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group  
19075 identifiers has not been allocated and this field shall be ignored.

#### 19076 **13.3.2.2.5.10 Free Network Address Range Begin Field**

19077 The *free network address range begin* field is 16-bits in length and shall contain the value of the  
19078 *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the recipient can  
19079 assign. If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent  
19080 allocation by the target.

#### 19081 **13.3.2.2.5.11 Free Network Address Range End Field**

19082 The *free network address range end* field is 16-bits in length and shall contain the value of the  
19083 *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the recipient can  
19084 assign. If the value of the *free network address range begin* field is equal to zero, a range of network addresses has  
19085 not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

#### 19086 **13.3.2.2.5.12 Free Group Identifier Range Begin Field**

19087 The *free group identifiers begin* field is 16-bits in length and shall contain the value of the  
19088 *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the recipient can  
19089 assign. If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent  
19090 allocation by the target.

#### 19091 **13.3.2.2.5.13 Free Group Identifier Range End Field**

19092 The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd*  
19093 attribute, specifying the end of the range of group identifiers that the recipient can assign. If the value of the *free group*  
19094 *identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator for  
19095 subsequent allocation by the target and this field shall be ignored.

#### 19096 **13.3.2.2.5.14 Initiator IEEE Address**

19097 The *initiator IEEE address* is 64-bits in length and shall contain the IEEE address of the initiator of the new network.

#### 19098 **13.3.2.2.5.15 Initiator Network Address Field**

19099 The *initiator network address* is 16-bits in length and shall contain the short network address of the initiator of the  
19100 new network.

### 19101 **13.3.2.2.6 Network Join Router Request Command Frame**

19102 The *network join router request* command frame is used by a non-factory-new initiator to join a router to its network.

19103 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
19104 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
19105 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
19106 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used  
19107 in the preceding scan request inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
19108 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
19109 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
19110 command identifier shall be set to 0x12 (network join router request).

19111 The ZCL payload field shall contain the network join router request command frame itself, formatted as illustrated in  
19112 Figure 13-15.

19113 **Figure 13-15. Format of the Network Join Router Request Command Frame**

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

- 19114 **13.3.2.2.6.1 Inter-PAN Transaction Identifier Field**
- 19115 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19116 This value shall be identical to the inter-PAN transaction identifier field of the original scan request inter-PAN  
19117 command frame sent by the initiator.
- 19118 **13.3.2.2.6.2 Extended PAN Identifier Field**
- 19119 The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.
- 19120 **13.3.2.2.6.3 Key Index Field**
- 19121 The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the  
19122 protection method) to be used in the *encrypted network key* field.
- 19123 **13.3.2.2.6.4 Encrypted Network Key Field**
- 19124 The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network,  
19125 encrypted according to the algorithm indicated by the *key index* field.
- 19126 **13.3.2.2.6.5 Network Update Identifier Field**
- 19127 The *network update identifier* field is 8-bits in length and shall specify the value of the *nwkUpdateId* attribute of the  
19128 initiator.
- 19129 **13.3.2.2.6.6 Logical Channel Field**
- 19130 The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.
- 19131 **13.3.2.2.6.7 PAN Identifier Field**
- 19132 The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.
- 19133 **13.3.2.2.6.8 Network Address Field**
- 19134 The *network address* field is 16-bits in length and contains the short network address assigned to the target.
- 19135 **13.3.2.2.6.9 Group Identifiers Begin Field**
- 19136 The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the  
19137 router can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.
- 19138 **13.3.2.2.6.10 Group Identifiers End Field**
- 19139 The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the router  
19140 can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group identifiers  
19141 has not been allocated and this field shall be ignored.
- 19142 **13.3.2.2.6.11 Free Network Address Range Begin Field**
- 19143 The *free network address range begin* field is 16-bits in length and shall contain the value of the  
19144 *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the router can assign.  
19145 If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent  
19146 allocation by the target.
- 19147 **13.3.2.2.6.12 Free Network Address Range End Field**
- 19148 The *free network address range end* field is 16-bits in length and shall contain the value of the  
19149 *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the router can assign.  
19150 If the value of the *free network address range begin* field is equal to zero, a range of network addresses has not been  
19151 allocated by the initiator for subsequent allocation by the target and this field shall be ignored.

19152 **13.3.2.2.6.13 Free Group Identifier Range Begin Field**

19153 The *free group identifiers begin* field is 16-bits in length and shall contain the value of the  
19154 *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the router can assign.  
19155 If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent allocation  
19156 by the target.

19157 **13.3.2.2.6.14 Free Group Identifier Range End Field**

19158 The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd*  
19159 attribute, specifying the end of the range of group identifiers that the router can assign. If the value of the *free group*  
19160 *identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator for  
19161 subsequent allocation by the target and this field shall be ignored.

19162 **13.3.2.2.7 Network Join End Device Request Command Frame**

19163 The *network join end device request* command frame is used by a non-factory-new initiator to join a factory new end  
19164 device to its network.

19165 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
19166 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
19167 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
19168 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
19169 the preceding *scan request* inter-PAN command frame, if the device is factory new, or the PAN identifier of the device,  
19170 otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
19171 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
19172 command identifier shall be set to 0x14 (network join end device request).

19173 The ZCL payload field shall contain the *network join end device request* command frame itself, formatted as illustrated  
19174 in Figure 13-16.

19175 **Figure 13-16. Format of the Network Join End Device Request Command Frame**

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

19176 **13.3.2.2.7.1 Inter-PAN Transaction Identifier Field**

19177 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19178 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
19179 command frame sent by the initiator.

- 19180 **13.3.2.2.7.2 Extended PAN Identifier Field**
- 19181 The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.
- 19182 **13.3.2.2.7.3 Key Index Field**
- 19183 The *key index* field is 8-bits in length and shall specify the index (in the range 0x00 – 0x0f) of the key (and hence the protection method) to be used in the *encrypted network key* field.
- 19184
- 19185 **13.3.2.2.7.4 Encrypted Network Key Field**
- 19186 The *encrypted network key* field is 128-bits in length and shall specify the network key to use for the network, encrypted according to the algorithm indicated by the *key index* field.
- 19187
- 19188 **13.3.2.2.7.5 Network Update Identifier Field**
- 19189 The *network update identifier* field is 8-bits in length and shall specify the current value of the *nwkUpdateId* attribute of the originator.
- 19190
- 19191 **13.3.2.2.7.6 Logical Channel Field**
- 19192 The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.
- 19193 **13.3.2.2.7.7 PAN Identifier Field**
- 19194 The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.
- 19195 **13.3.2.2.7.8 Network Address Field**
- 19196 The *network address* field is 16-bits in length and contains the short network address assigned to the target.
- 19197 **13.3.2.2.7.9 Group Identifiers Begin Field**
- 19198 The *group identifiers begin* field is 16-bits in length and specifies the start of the range of group identifiers that the end device can use for its endpoints. If this value is equal to zero, a range of group identifiers has not been allocated.
- 19199
- 19200 **13.3.2.2.7.10 Group Identifiers End Field**
- 19201 The *group identifiers end* field is 16-bits in length and specifies the end of the range of group identifiers that the end device can use for its endpoints. If the value of the *group identifiers begin* field is equal to zero, a range of group identifiers has not been allocated and this field shall be ignored.
- 19202
- 19203
- 19204 **13.3.2.2.7.11 Free Network Address Range Begin Field**
- 19205 The *free network address range begin* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeBegin* attribute, specifying the start of the range of network addresses that the end device can assign. If this value is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target.
- 19206
- 19207
- 19208
- 19209 **13.3.2.2.7.12 Free Network Address Range End Field**
- 19210 The *free network address range end* field is 16-bits in length and shall contain the value of the *aplFreeNwkAddrRangeEnd* attribute, specifying the end of the range of network addresses that the end device can assign. If the value of the *free network address range begin* field is equal to zero, a range of network addresses has not been allocated by the initiator for subsequent allocation by the target and this field shall be ignored.
- 19211
- 19212
- 19213
- 19214 **13.3.2.2.7.13 Free Group Identifier Range Begin Field**

19215 The *free group identifiers begin* field is 16-bits in length and shall contain the value of the  
19216 *aplFreeGroupIDRangeBegin* attribute, specifying the start of the range of group identifiers that the end device can  
19217 assign. If this value is equal to zero, a range of group identifiers has not been allocated by the initiator for subsequent  
19218 allocation by the target.

#### 19219 **13.3.2.2.7.14 Free Group Identifier Range End Field**

19220 The *free group identifiers end* field is 16-bits in length and shall contain the value of the *aplFreeGroupIDRangeEnd*  
19221 attribute, specifying the end of the range of group identifiers that the end device can assign. If the value of the *free*  
19222 *group identifier range begin* field is equal to zero, a range of group identifiers has not been allocated by the initiator  
19223 for subsequent allocation by the target and this field shall be ignored.

### 19224 **13.3.2.2.8 Network Update Request Command Frame**

19225 The *network update request* command frame is used to attempt to bring a router that may have missed a network  
19226 update back onto the network.

19227 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
19228 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
19229 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
19230 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the PAN identifier of the  
19231 initiating device. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00 (normal  
19232 unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 0 (client to server) and the  
19233 command identifier shall be set to 0x16 (network update request).

19234 The ZCL payload field shall contain the *network update request* command frame itself, formatted as in Figure 13-17.

19235 **Figure 13-17. Format of the Network Update Request Command Frame**

<b>Octets</b>	4	8	1	1	2	2
<b>Data Type</b>	uint32	IEEE address	uint8	uint8	uint16	uint16
<b>Field Name</b>	Inter-PAN transaction identifier	Extended PAN identifier	Network update identifier	Logical channel	PAN identifier	Network address

#### 19236 **13.3.2.2.8.1 Inter-PAN Transaction Identifier Field**

19237 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19238 This field shall contain a non-zero 32-bit random number and is used to identify the current network update request.

#### 19239 **13.3.2.2.8.2 Extended PAN Identifier Field**

19240 The *extended PAN identifier* field is 64-bits in length and shall contain the extended PAN identifier of the network.

#### 19241 **13.3.2.2.8.3 Network Update Identifier Field**

19242 The *network update identifier* field is 8-bits in length and shall specify the current value of the *nwkUpdateId* attribute  
19243 of the originator.

#### 19244 **13.3.2.2.8.4 Logical Channel Field**

19245 The *logical channel* field is 8-bits in length and shall contain the ZLL channel to be used for the network.

#### 19246 **13.3.2.2.8.5 PAN Identifier Field**

19247 The *PAN identifier* field is 16-bits in length and shall contain the PAN identifier of the network.

19248 **13.3.2.2.8.6 Network Address Field**

19249 The *network address* field is 16-bits in length and contains the short network address assigned to the target.

19250 **13.3.2.2.9 Get Group Identifiers Request Command**

19251 The *get group identifiers request* command is used to retrieve the actual group identifiers that the endpoint is using in  
 19252 its multicast communication in controlling different (remote) devices.

19253 This command shall be formatted as illustrated in Figure 13-18.

19254 **Figure 13-18. Format of the Get Group Identifiers Request Command**

<b>Octets</b>	1
<b>Data Type</b>	Unsigned 8-bit integer
<b>Field Name</b>	Start index

19255 **13.3.2.2.9.1 Start Index Field**

19256 The *start index* field is 8-bits in length and shall contain the index (starting from 0) at which to start returning group  
 19257 identifiers.

19258 **13.3.2.2.10 Get Endpoint List Request Command**

19259 The *get endpoint list request* command is used to retrieve addressing information for each endpoint the device is using  
 19260 in its unicast communication in controlling different (remote) devices.

19261 This command shall be formatted as illustrated in Figure 13-19.

19262 **Figure 13-19. Format of the Get Endpoint List Request Command**

<b>Octets</b>	1
<b>Data Type</b>	Unsigned 8-bit integer
<b>Field Name</b>	Start index

19263 **13.3.2.2.10.1 Start Index Field**

19264 The *start index* field is 8-bits in length and shall contain the index (starting from 0) at which to start returning endpoint  
 19265 identifiers.

19266 **13.3.2.3 Commands Generated**

19267 When a device implements the *touchlink commissioning* cluster at the ZCL server side, it shall be able to generate the  
 19268 commands listed in Table 13-15.

19269

**Table 13-15. Commands Generated by the Server Side of the Touchlink Commissioning Cluster**

	Command Identifier Field Value	Description	Mandatory/Optional	Reference
Touchlink	0x01	Scan response	Mandatory	13.3.2.3.1
	0x03	Device information response	Mandatory	13.3.2.3.2
	0x11	Network start response	Mandatory	13.3.2.3.3
	0x13	Network join router response	Mandatory	13.3.2.3.4
	0x15	Network join end device response	Mandatory	13.3.2.3.5
	All other values in the range 0x00 – 0x3f	Reserved	-	-
Utility	0x40	Endpoint information	Mandatory	13.3.2.3.6
	0x41	Get group identifiers response	Mandatory	13.3.2.3.7
	0x42	Get endpoint list response	Mandatory	13.3.2.3.8
	All other values in the range 0x40 – 0xff	Reserved	-	-

19270 **13.3.2.3.1 Scan Response Command Frame**

19271 The *scan response* command frame is used to respond to the originator of a *scan request* command frame with device  
 19272 details. The information contained in this command frame relates to the target that is responding to the scan request  
 19273 command frame.

19274 Note: If the Profile Interop bit of the Touchlink Information field of the received Scan Request command is  
 19275 set to zero, the device may choose to represent its device information in the form of ZLL device  
 19276 information to support legacy devices. If this bit is set to one, the device shall use the device information  
 19277 as given in its simple descriptors.<sup>158</sup>

19278 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 19279 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 19280 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 19281 shall contain the IEEE address of the destination and the source PAN ID field shall be set to any value in the range  
 19282 0x0001 – 0xffff, if the device is factory new, or the PAN identifier of the device, otherwise. In the APS header, the  
 19283 delivery mode sub-field of the frame control field shall be set to 0b00 (normal unicast). In the ZCL header, the direction  
 19284 sub-field of the frame control field shall be set to 1 (server to client) and the command identifier shall be set to 0x01  
 19285 (scan response).

19286 The ZCL payload field shall contain the scan response command frame itself, formatted as illustrated in Figure 13-20.

19287

**Figure 13-20. Format of the Scan Response Command Frame**

Octets	Data Type	Field Name
4	Unsigned 32-bit integer	Inter-PAN transaction identifier
1	Unsigned 8-bit integer	RSSI correction
1	8-bit bitmap	ZigBee information

<sup>158</sup> CCB 2115



Octets	Data Type	Field Name
1	8-bit bitmap	Touchlink information
2	16-bit bitmap	Key bitmask
4	Unsigned 32-bit integer	Response identifier
8	IEEE address	Extended PAN identifier
1	Unsigned 8-bit integer	Network update identifier
1	Unsigned 8-bit integer	Logical channel
2	Unsigned 16-bit integer	PAN identifier
2	Unsigned 16-bit integer	Network address
1	Unsigned 8-bit integer	Number of sub-devices
1	Unsigned 8-bit integer	Total group identifiers
0/1	Unsigned 8-bit integer	Endpoint identifier
0/2	Unsigned 16-bit integer	Profile identifier
0/2	Unsigned 16-bit integer	Device identifier
0/1	Unsigned 8-bit integer	Version
0/1	Unsigned 8-bit integer	Group identifier count

19288 **13.3.2.3.1.1 Inter-PAN Transaction Identifier Field**

19289 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 19290 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
 19291 command frame received from the initiator.

19292 **13.3.2.3.1.2 RSSI Correction Field**

19293 The *RSSI correction* field is 8-bits in length and specifies a pre-programmed RSSI correction offset, specific to this  
 19294 device in the range 0x00 – 0x20.

19295 **13.3.2.3.1.3 ZigBee Information Field**

19296 The *ZigBee information* field is 8-bits in length and specifies information related to ZigBee. This field shall be  
 19297 formatted as illustrated in Figure 13-21.

19298 **Figure 13-21. Format of the ZigBee Information Field**

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

19299 The *logical type* subfield is 2-bits in length and specifies the logical type of the device. The value of this subfield shall  
 19300 be set to 0b00 for a coordinator, 0b01 for a router or 0b10 for an end device.

19301 The Rx on when idle subfield is 1 bit in length and specifies the *RxOnWhenIdle* state of the device. The value of this  
 19302 subfield shall be set to 1 to indicate that the receiver is left on when the device is idle or 0 otherwise.

19303 **13.3.2.3.1.4 Touchlink Information Field**

19304 The *Touchlink information* field is 8-bits in length and shall be formatted as illustrated in Figure 13-22.

19305

**Figure 13-22. Format of the Scan Response Touchlink Information Field**

Bits: 0	1	2-3	4	5	6	7
Factory new	Address assignment	Reserved	Touchlink initiator	Touchlink priority request	Reserved	Profile Interop <sup>159</sup>

19306

19307 The *factory new* subfield is 1 bit in length and specifies whether the device is factory new. The value of this subfield  
19308 shall be set to 1 to indicate the device is factory new or 0 otherwise.

19309 The address assignment subfield is 1 bit in length and specifies whether the device is capable of assigning addresses.  
19310 The value of this subfield shall be set to 1 to indicate the device is capable of assigning addresses or 0 otherwise.

19311 The touchlink initiator subfield is 1 bit in length and specifies whether the device is initiating a touchlink operation.  
19312 The value of this subfield shall be set to 1 to indicate the device is initiating a touchlink or 0 otherwise.

19313 The *touchlink priority request* subfield is 1 bit in length and specifies that the target has requested some priority,  
19314 possibly after a button push by the user. The value of this subfield shall be set to 1 to indicate that the device has  
19315 requested priority or 0 otherwise.

19316 The Profile Interop subfield is 1 bit in length and specifies which profile the device implements. If the ZLL profile is  
19317 implemented, this bit shall be set to 0. In all other case (Profile Interop / ZigBee 3.0), this bit shall be set to 1.<sup>160</sup>

#### 19318 **13.3.2.3.1.5 Key Bitmask Field**

19319 The *key bitmask* field is 16-bits in length and specifies which keys (and hence which encryption algorithms) are  
19320 supported by the device. The appropriate key shall be present on the device only if its corresponding bit is set to 1  
19321 otherwise the key is not supported. Bit  $i$  of the *key bitmask field* shall correspond to key index  $i$ , where  $0 \leq i \leq 15$ .

#### 19322 **13.3.2.3.1.6 Response Identifier Field**

19323 The *response identifier* field is 32-bits in length and specifies a random identifier for the response, used during the  
19324 network key transfer mechanism.

#### 19325 **13.3.2.3.1.7 Extended PAN Identifier Field**

19326 The *extended PAN identifier* field is 64-bits in length and specifies the extended PAN identifier of the device  
19327 responding to the scan request.

19328 If the *factory new* subfield of the *touchlink information* field indicates that the device is factory new and the value of  
19329 this field is equal to zero, the target is not able to propose any network parameters. If the *factory new* subfield of the  
19330 *touchlink information* field indicates that the device is factory new and the value of this field is not equal to zero, it  
19331 can be used as the extended PAN identifier of a potential new network. Alternatively, if the *factory new* subfield of  
19332 the *touchlink information* field indicates that the device is not factory new, this field indicates the current extended  
19333 PAN identifier of the network on which the device operates.

#### 19334 **13.3.2.3.1.8 Network Update Identifier Field**

19335 The network update identifier field is 8-bits in length and specifies the current value of the *nwkUpdateId* attribute of  
19336 the originator. If the factory new subfield of the touchlink information indicates the device to be in factory new mode,  
19337 this field shall contain the value 0x00.

#### 19338 **13.3.2.3.1.9 Logical Channel Field**

19339 The logical channel field is 8-bits in length and specifies the touchlink channel on which the device is operating.

<sup>159</sup> CCB 2115<sup>160</sup> CCB 2115

19340 If the *factory new* subfield of the *touchlink information* field indicates that the device is factory new and the value of  
19341 the extended PAN identifier field is equal to zero, the target is not able to propose a logical channel for the network.  
19342 If the *factory new* subfield of the *touchlink information* field indicates that the device is factory new and the value of  
19343 the extended PAN identifier field is not equal to zero, this value can be used as the logical channel of a potential new  
19344 network. Alternatively, if the *factory new* subfield of the *touchlink information* field indicates that the device is not  
19345 factory new, this field indicates the current logical channel of the network on which the device operates.

#### 19346 **13.3.2.3.1.10 PAN Identifier Field**

19347 The PAN identifier field is 16-bits in length and specifies the identifier of the PAN on which the device operates.

19348 If the *factory new* subfield of the *touchlink information* field indicates that the device is factory new and the value of  
19349 the extended PAN identifier field is equal to zero, the target is not able to propose a PAN identifier for the network.  
19350 If the *factory new* subfield of the *touchlink information* field indicates that the device is factory new and the value of  
19351 the extended PAN identifier field is not equal to zero, this value can be used as the PAN identifier of a potential new  
19352 network. Alternatively, if the *factory new* subfield of the *touchlink information* field indicates that the device is not  
19353 factory new, this field indicates the current PAN identifier of the network on which the device operates.

#### 19354 **13.3.2.3.1.11 Network Address Field**

19355 The network address field is 16-bits in length and specifies the current network address of the device. If the factory  
19356 new subfield of the touchlink information indicates the device to be in factory new mode, this value shall be set to  
19357 0xffff.

#### 19358 **13.3.2.3.1.12 Number of Sub-devices Field**

19359 The number of sub-devices field is 8-bits in length and specifies the number of sub-devices (endpoints) supported by  
19360 the device.

#### 19361 **13.3.2.3.1.13 Total Group Identifiers Field**

19362 The total group identifiers field is 8-bits in length and specifies the number of unique group identifiers that this device  
19363 requires.

#### 19364 **13.3.2.3.1.14 Endpoint Identifier Field**

19365 The endpoint identifier field is 8-bits in length and specifies the endpoint identifier of the sub-device. This field shall  
19366 only be present when the number of sub-devices field is equal to 1.

#### 19367 **13.3.2.3.1.15 Profile Identifier Field**

19368 The profile identifier field is 16-bits in length and specifies the profile identifier supported by the sub-device. This  
19369 field shall only be present when the number of sub-devices field is equal to 1.

#### 19370 **13.3.2.3.1.16 Device Identifier Field**

19371 The device identifier field is 16-bits in length and specifies the device identifier supported by the sub-device. This  
19372 field shall only be present when the number of sub-devices field is equal to 1.

#### 19373 **13.3.2.3.1.17 Version Field**

19374 The version field is 8-bits in length and specifies the version of the device description supported by the sub-device on  
19375 the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to the  
19376 *application device version* field of the appropriate simple descriptor; the most significant 4 bits shall be set to 0x0.

19377 This field shall only be present when the number of sub-devices field is equal to 1.

#### 19378 **13.3.2.3.1.18 Group Identifier Count Field**

19379 The group identifier count field is 8-bits in length and specifies the number of group identifiers required by the sub-  
19380 device. This field shall only be present when the number of sub-devices field is equal to 1.

19381 **13.3.2.3.2 Device Information Response Command Frame**

19382 The *device information response* command frame is used to return information about the sub-devices supported by a  
19383 node.

19384 Note: If the Profile Interop bit of the Touchlink Information field of the Scan Request command received at the  
19385 beginning of the current touchlink exchange is set to zero, the device may choose to represent its device information  
19386 in the form of ZLL device information to support legacy devices. If this bit is set to one, the device shall use the device  
19387 information as given in its simple descriptors.<sup>161</sup>

19388 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
19389 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
19390 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
19391 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
19392 the preceding scan response inter-PAN command frame, if the device is factory new, or the PAN identifier of the  
19393 device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00  
19394 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client)  
19395 and the command identifier shall be set to 0x03 (device information response).

19396 The ZCL payload field shall contain the *device information response* command frame itself, formatted as illustrated  
19397 in Figure 13-23.

19398 **Figure 13-23. Format of the Device Information Response Command Frame**

<b>Octets</b>	4	1	1	1	(n*16)
<b>Data Type</b>	uint32	uint8	uint8	uint8	Variable
<b>Field Name</b>	Inter-PAN transaction identifier	Number of sub devices	Start index	Device information record count	Device information record (see Figure 13-24)

19399

19400 **Figure 13-24. Format of the Device Information Record Field**

<b>Octets</b>	8	1	2	2	1	1	1
<b>Data Type</b>	IEEE address	uint8	uint16	uint16	uint8	uint8	uint8
<b>Field Name</b>	IEEE address	Endpoint identifier	Profile identifier	Device identifier	Version	Group identifier count	Sort

19401 **13.3.2.3.2.1 Inter-PAN Transaction Identifier Field**

19402 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19403 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
19404 command frame received from the initiator.

19405 **13.3.2.3.2.2 Number of Sub-devices Field**

<sup>161</sup> CCB 2115

19406 The *number of sub devices* field is 8-bits in length and specifies the number of sub devices contained in the device, as  
19407 reported in the *scan response* inter-PAN command frame.

#### 19408 **13.3.2.3.2.3 Start Index Field**

19409 The *start index* field is 8-bits in length and specifies the starting index into the device table from which to get device  
19410 information. This value of this field shall be equal to the value of the start index field of the *device information request*  
19411 command frame.

#### 19412 **13.3.2.3.2.4 Device Information Record Count Field**

19413 The *device information record count* field is 8-bits in length and specifies the number *n* of device information records  
19414 that follow. This value shall be in the range 0x00 – 0x05.

#### 19415 **13.3.2.3.2.5 IEEE Address Field**

19416 The *IEEE address* field is 64-bits in length and shall contain the IEEE address of the device referred to by the device  
19417 information record.

#### 19418 **13.3.2.3.2.6 Endpoint Identifier Field**

19419 The *endpoint identifier* field is 8-bits in length and shall contain the endpoint identifier of the sub-device referred to  
19420 by the device information record.

#### 19421 **13.3.2.3.2.7 Profile Identifier**

19422 The *profile identifier* field is 16-bits in length and shall contain the identifier of the profile supported by the sub-device  
19423 referred to by the device information record.

#### 19424 **13.3.2.3.2.8 Device Identifier Field**

19425 The *device identifier* field is 16-bits in length and shall contain the device identifier of the sub-device referred to by  
19426 the device information record.

#### 19427 **13.3.2.3.2.9 Version Field**

19428 The *version* field is 8-bits in length and shall contain the version of the device description supported by the sub-device  
19429 on the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to  
19430 the *application device version* field of the appropriate simple descriptor; the most significant 4 bits shall be set to 0x0.

#### 19431 **13.3.2.3.2.10 Group Identifier Count Field**

19432 The *group identifier count* field is 8-bits in length and shall contain the number of unique group identifiers required  
19433 by the sub-device referred to by the device information record.

#### 19434 **13.3.2.3.2.11 Sort Field**

19435 The *sort* field is 8-bits in length and shall contain the sorting order of the sub-device referred to by the device  
19436 information record. This field is used to identify if a sorting of sub-devices is needed and what the order is, e.g., to  
19437 sort the different lights in a luminaire in a selection list on the remote control. A value of zero shall indicate ‘not  
19438 sorted’. Non-zero values shall indicate the order in the list, with the value 0x01 indicating the top of the list.

### 19439 **13.3.2.3.3 Network Start Response Command Frame**

19440 The *network start response* command frame is used by a router to respond to a *network start request* command frame  
19441 received from an end device.

19442 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 19443 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 19444 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 19445 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 19446 the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the  
 19447 device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00  
 19448 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client)  
 19449 and the command identifier shall be set to 0x11 (network start response).

19450 The ZCL payload field shall contain the *network start response* command frame itself, formatted as illustrated in  
 19451 Figure 13-25.

19452 **Figure 13-25. Format of the Network Start Response Command Frame**

Octets	4	1	8	1	1	2
Data Type	uint32	uint8	IEEE address	uint8	uint8	uint16
Field Name	Inter-PAN transaction identifier	Status	Extended PAN identifier	Network update identifier	Logical channel	PAN identifier

#### 19453 **13.3.2.3.3.1 Inter-PAN Transaction Identifier Field**

19454 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 19455 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
 19456 command frame received from the initiator.

#### 19457 **13.3.2.3.3.2 Status Field**

19458 The status field is 8-bits in length and shall contain the status code corresponding to the result of the network start  
 19459 request. This field shall be set to one of the values listed in Table 13-16.

19460 **Table 13-16. Values of the Status Field of the Network Start Response Command Frame**

Status Field Value	Description
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

#### 19461 **13.3.2.3.3.3 Extended PAN Identifier Field**

19462 The *extended PAN identifier* field is 64-bits in length and shall contain the extended identifier of the new PAN.

#### 19463 **13.3.2.3.3.4 Network Update Identifier Field**

19464 The *network update identifier* field is 8-bits in length and shall be set to 0x00 in this version of the specification.

#### 19465 **13.3.2.3.3.5 Logical Channel Field**

19466 The *logical channel* field is 8-bits in length and shall contain the ZLL channel used by the new network.

#### 19467 **13.3.2.3.3.6 PAN Identifier Field**

19468 The *PAN identifier* field is 16-bits in length and shall contain the identifier of the new PAN.

19469 **13.3.2.3.4 Network Join Router Response Command Frame**

19470 The *network join router response* command frame is used by a router to respond to a *network join router request*  
 19471 command frame received from a non-factory-new end device.

19472 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
 19473 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
 19474 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
 19475 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
 19476 the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the  
 19477 device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00  
 19478 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client)  
 19479 and the command identifier shall be set to 0x13 (network join router response).

19480 The ZCL payload field shall contain the *network join router response* command frame itself, formatted as illustrated  
 19481 in Figure 13-26.

19482 **Figure 13-26. Format of the Network Join Router Response Command Frame**

<b>Octets</b>	4	1
<b>Data Type</b>	Unsigned 32-bit integer	Unsigned 8-bit integer
<b>Field Name</b>	Inter-PAN transaction identifier	Status

19483 **13.3.2.3.4.1 Inter-PAN Transaction Identifier Field**

19484 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
 19485 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
 19486 command frame received from the initiator.

19487 **13.3.2.3.4.2 Status Field**

19488 The *status* field is 8-bits in length and shall contain the status code corresponding to the result of the network join  
 19489 router request. This field shall be set to one of the values listed in Table 13-17.

19490 **Table 13-17. Values of the Status Field of the Network Join Router Response Command Frame**

Status Field Value	Description
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

19491 **13.3.2.3.5 Network Join End Device Response Command Frame**

19492 The *network join end device response* command frame is used by a factory new end device to respond to a *network*  
 19493 *join end device request* command frame received from a non-factory new end device.

19494 This inter-PAN command shall be formatted according to the general inter-PAN frame format with the following  
19495 clarifications. In the MAC header, the ACK request and destination addressing mode sub-fields of the frame control  
19496 field shall be set to 1 (ACK requested) and 0b11 (extended IEEE address), respectively, the destination address field  
19497 shall contain the IEEE address of the destination and the source PAN ID field shall be set to the same value used in  
19498 the preceding *scan response* inter-PAN command frame, if the device is factory new, or the PAN identifier of the  
19499 device, otherwise. In the APS header, the delivery mode sub-field of the frame control field shall be set to 0b00  
19500 (normal unicast). In the ZCL header, the direction sub-field of the frame control field shall be set to 1 (server to client)  
19501 and the command identifier shall be set to 0x15 (network join end device response).

19502 The ZCL payload field shall contain the *network join end device response* command frame itself, formatted as  
19503 illustrated in Figure 13-27.

19504 **Figure 13-27. Format of the Network Join End Device Response Command Frame**

<b>Octets</b>	4	1
<b>Data Type</b>	Unsigned 32-bit integer	Unsigned 8-bit integer
<b>Field Name</b>	Inter-PAN transaction identifier	Status

19505 **13.3.2.3.5.1 Transaction Identifier Field**

19506 The *inter-PAN transaction identifier* field is 32-bits in length and specifies an identifier for the inter-PAN transaction.  
19507 This value shall be identical to the *inter-PAN transaction identifier* field of the original *scan request* inter-PAN  
19508 command frame received from the initiator.

19509 **13.3.2.3.5.2 Status Field**

19510 The *status* field is 8-bits in length and shall contain the status code corresponding to the result of the network join end  
19511 device request. This field shall be set to one of the values listed in Table 13-18.



19512 **Table 13-18. Values of the Status Field of the Network Join End Device Response Command Frame**

Status Field Value	Description
0x00	Success
0x01	Failure
0x02 – 0xff	Reserved

19513 **13.3.2.3.6 Endpoint Information Command**

19514 The *endpoint information* command is used to inform the remote endpoint about the general information of the local  
 19515 endpoint. This command may be a trigger for the remote endpoint to get more information from the local device using  
 19516 the other commands described in this cluster.

19517 **Note:** if the related endpoint(s) reside on sleeping end devices, the polling time and polling frequency must be chosen  
 19518 such that the exchange of information is done efficiently and in a timely manner.

19519 The endpoint information command shall be sent using unicast transmission. On receipt of this command, the device  
 19520 shall respond using a ZCL default response command.

19521 This command shall be formatted as illustrated in Figure 13-28.

19522 **Figure 13-28. Format of the Endpoint Information Command**

Octets	8	2	1	2	2	1
Data Type	IEEE address	uint16	uint8	uint16	uint16	uint8
Field Name	IEEE address	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

19523 **13.3.2.3.6.1 IEEE Address Field**

19524 The *IEEE address* field is 64-bits in length and specifies the IEEE address of the local device.

19525 **13.3.2.3.6.2 Network Address Field**

19526 The *network address* field is 16-bits in length and specifies the short network address of the local device.

19527 **13.3.2.3.6.3 Endpoint Identifier Field**

19528 The *endpoint identifier* field is 8-bits in length and specifies the identifier of the local endpoint.

19529 **13.3.2.3.6.4 Profile Identifier Field**

19530 The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported on the endpoint  
 19531 specified in the *endpoint identifier* field.

19532 **13.3.2.3.6.5 Device Identifier Field**

19533 The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported on the  
 19534 endpoint specified in the *endpoint identifier* field.

19535 **13.3.2.3.6.6 Version Field**

19536 The *version* field is 8-bits in length and specifies the version of the device description supported by the sub-device on  
 19537 the endpoint specified by the *endpoint identifier* field. The least significant 4 bits of this value shall correspond to the  
 19538 *application device version* field of the appropriate simple descriptor; the most significant 4 bits shall be set to 0x0.

19539 **13.3.2.3.7 Get Group Identifiers Response Command**

19540 The *get group identifiers response* command allows a remote device to respond to the get group identifiers request  
 19541 command.

19542 This command shall be formatted as illustrated in Figure 13-29.

19543 **Figure 13-29. Format of the Get Group Identifiers Response Command**

<b>Octets</b>	1	1	1	(n*3)
<b>Data Type</b>	uint8	uint8	uint8	Variable
<b>Field Name</b>	Total	Start index	Count	Group information record list

19544 **13.3.2.3.7.1 Total Field**

19545 The *total* field is 8-bits in length and specifies the total number of group identifiers supported by the device.

19546 **13.3.2.3.7.2 Start Index Field**

19547 The *start index* field is 8-bits in length and specifies the internal starting index from which the following group  
 19548 identifiers are taken and corresponds to the *start index* field of the *get group identifiers request* command.

19549 **13.3.2.3.7.3 Count Field**

19550 The *count* field is 8-bits in length and specifies the number of entries in the *group information record list* field. If no  
 19551 entries are returned, this field shall be set to 0.

19552 **13.3.2.3.7.4 Group Information Record List Field**

19553 The *group information record* field is ( $n * 24$ )-bits in length, where  $n$  is equal to the value of the *count* field, and  
 19554 specifies the requested group information. Each entry in this field shall be formatted as illustrated in Figure 13-30.

19555 **Figure 13-30. Format of a Group Information Record Entry**

<b>Octets</b>	2	1
<b>Data Type</b>	uint16	uint8
<b>Field Name</b>	Group identifier	Group type

19556 The *group identifier* sub-field is 16-bits in length and specifies the identifier of the group described by this record.  
 19557 The *group type* sub-field is 8-bits in length and has been introduced for future extensions. The group type shall indicate

19558 the meaning of a group in the user interface. In the current version of this specification, this value shall be set to 0x00.  
 19559

19560 **13.3.2.3.8 Get Endpoint List Response Command**

19561 The *get endpoint list response* command allows a remote device to respond to the get endpoint list request command.  
 19562 This command shall be formatted as illustrated in Figure 13-31.

19563 **Figure 13-31. Format of the Get Endpoint List Response Command**

<b>Octets</b>	1	1	1	(n*8)
<b>Data Type</b>	uint8	uint8	uint8	Variable
<b>Field Name</b>	Total	Start index	Count	Endpoint information record list (see Figure 13-32)

19564

19565 **Figure 13-32. Format of an Endpoint Information Record Entry**

<b>Octets</b>	2	1	2	2	1
<b>Data Type</b>	uint16	uint8	uint16	uint16	uint8
<b>Field Name</b>	Network address	Endpoint identifier	Profile identifier	Device identifier	Version

19566 **13.3.2.3.8.1 Total Field**

19567 The *total* field is 8-bits in length and specifies the total number of endpoints supported by the device.

19568 **13.3.2.3.8.2 Start Index Field**

19569 The *start index* field is 8-bits in length and specifies the internal starting index from which the following list of  
 19570 endpoints are taken and corresponds to the *start index* field of the *get endpoint list request* command.

19571 **13.3.2.3.8.3 Count Field**

19572 The *count* field is 8-bits in length and specifies the number of entries in the *endpoint information record list* field. If  
 19573 no entries are returned, this field shall be set to 0.

19574 **13.3.2.3.8.4 Network Address Field**

19575 The *network address* field is 16-bits in length and specifies the short network address of the device specified by the  
 19576 current endpoint information record.

19577 **13.3.2.3.8.5 Endpoint Identifier Field**

19578 The *endpoint identifier* field is 8-bits in length and specifies the identifier of the endpoint on the device specified by  
 19579 the *network address* field.

19580 **13.3.2.3.8.6 Profile Identifier Field**

19581 The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported on the endpoint,  
 19582 specified in the *endpoint identifier* field, on the device specified by the *network address* field.

19583 **13.3.2.3.8.7 Device Identifier Field**

19584 The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported on the  
19585 endpoint, specified in the *endpoint identifier* field, on the device specified by the *network address* field.

#### 19586 **13.3.2.3.8.8 Version Field**

19587 The *version* field is 8-bits in length and specifies the version of the device description supported by the sub-device on  
19588 the endpoint, specified by the *endpoint identifier* field, on the device specified by the *network address* field. The least  
19589 significant 4 bits of this value shall correspond to the *application device version* field of the appropriate simple  
19590 descriptor; the most significant 4 bits shall be set to 0x0.

### 19591 **13.3.3 Client**

#### 19592 **13.3.3.1 Attributes**

19593 The client has no attributes.

#### 19594 **13.3.3.2 Commands Received**

19595 The client receives the cluster specific response commands listed in Table 13-19. These commands are detailed in  
19596 13.3.2.2.

19597 **Table 13-19. Commands Received by the Client Side of the ZLL Commissioning Cluster**

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

#### 19598 **13.3.3.3 Commands Generated**

19599 The client generates the cluster specific commands listed in Table 13-20. These commands are detailed in 13.3.2.3.

19600 **Table 13-20. Commands Generated by the Client Side of the ZLL Commissioning Cluster**

	Identifier	Description	Usage
Touchlink	0x00	Scan request	Mandatory
	0x02	Device information request	Mandatory
	0x06	Identify request	Mandatory

	Identifier	Description	Usage
	0x07	Reset to factory new request	Mandatory
	0x10	Network start request	Mandatory
	0x12	Network join router request	Mandatory
	0x14	Network join end device request	Mandatory
	0x16	Network update request	Mandatory
Utility	0x41	Get group identifiers request	Optional
	0x42	Get endpoint list request	Optional

19601 **13.3.4 Functional Description**

19602 **13.3.4.1 Profile Identifier**

19603 Those commands in the touchlink commissioning command set shall be sent using the profile identifier, 0xc05e  
 19604 whereas those commands in the commissioning utility command set shall sent using the ZHA profile identifier,  
 19605 0x0104.

19606 **13.3.4.2 Constants**

19607 The constants that define the characteristics of touchlink commissioning are listed in Table 13-21.

19608 **Table 13-21. Touchlink Commissioning Constants**

Constant	Description	Value
<i>aplInterPANTransIdLifetime</i>	The maximum length of time an inter-PAN transaction identifier remains valid.	8s
<i>aplMinStartupDelayTime</i>	The length of time an initiator waits to ensure that the recipient has completed its startup procedure.	2s
<i>aplRxWindowDuration</i>	The maximum duration that a device leaves its receiver enabled during the joining procedure for subsequent configuration information.	5s
<i>aplScanTimeBaseDuration</i>	The base duration for a scan operation during which the receiver is enabled for scan responses.	0.25s

19609 **13.3.4.3 Attributes**

19610 Touchlink commissioning defines internal attributes required to allow a device to manage the way it operates. These  
 19611 attributes are summarized in Table 13-22.

19612 **Table 13-22. Touchlink Commissioning Attributes**

Attribute	Type	Ref	Default
<i>aplFreeNwkAddrRangeBegin</i>	Unsigned 16-bit integer	10.1.1.3.1	0x0001

<i>aplFreeNwkAddrRangeEnd</i>	Unsigned 16-bit integer	10.1.1.3.2	0xfff7
<i>aplFreeGroupIDRangeBegin</i>	Unsigned 16-bit integer	10.1.1.3.3	0x0001
<i>aplFreeGroupIDRangeEnd</i>	Unsigned 16-bit integer	10.1.1.3.4	0xfeff

#### 19613 13.3.4.3.1 **aplFreeNwkAddrRangeBegin Attribute**

19614 The *aplFreeNwkAddrRangeBegin* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfff7 and contains  
 19615 the starting value of the free network address range for address assignment capable devices. Address assignment  
 19616 capable devices should use and maintain this value when assigning addresses via touchlink commissioning. If the  
 19617 device is not address assignment capable or it has joined a network through classical ZigBee joining mechanisms, this  
 19618 attribute should be set to 0x0000.

#### 19619 13.3.4.3.2 **aplFreeNwkAddrRangeEnd Attribute**

19620 The *aplFreeNwkAddrRangeEnd* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfff7 and contains the  
 19621 end value of the free network address range for address assignment capable devices. Address assignment capable  
 19622 devices should use and maintain this value when assigning addresses via touchlink commissioning. If the device is  
 19623 not address assignment capable or it has joined a network through classical ZigBee joining mechanisms, this attribute  
 19624 should be set to 0x0000.

#### 19625 13.3.4.3.3 **aplFreeGroupIDRangeBegin Attribute**

19626 The *aplFreeGroupIDRangeBegin* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfeff and contains the  
 19627 starting value of the free group identifier range for address assignment capable devices. Address assignment capable  
 19628 devices should use and maintain this value when assigning group identifiers via touchlink commissioning. If the device  
 19629 is not address assignment capable or it has joined a network through classical ZigBee joining mechanisms, this  
 19630 attribute should be set to 0x0000.

#### 19631 13.3.4.3.4 **aplFreeGroupIDRangeEnd Attribute**

19632 The *aplFreeGroupIDRangeEnd* attribute is an unsigned 16-bit integer in the range 0x0000 – 0xfeff and contains the  
 19633 end value of the free group identifier range for address assignment capable devices. Address assignment capable  
 19634 devices should use and maintain this value when assigning group identifiers via touchlink commissioning. If the device  
 19635 is not address assignment capable or it has joined a network through classical ZigBee joining mechanisms, this  
 19636 attribute should be set to 0x0000.

### 19637 13.3.4.4 **Device Information Table**

19638 Each device supporting touchlink commissioning shall contain a *device information table* that holds the necessary  
 19639 (static) application information that is exchanged during touchlink device discovery. Each entry gives information  
 19640 about a so called sub-device which is a self-contained device such as a dimmable light. In ZigBee terms, a sub-device  
 19641 resides as a device application on an endpoint. Each entry shall be formatted as illustrated in Figure 13-33.

19642

Figure 13-33. Format of the device information table

Field name	Data type	Bits
IEEE address	IEEE address	64
Endpoint identifier	Unsigned 8-bit integer	8
Profile identifier	Unsigned 16-bit integer	16
Device identifier	Unsigned 16-bit integer	16

Device version	Unsigned 4-bit integer	4
Reserved	-	4
Number of groups identifiers	Unsigned 8-bit integer	8
Sort tag	Unsigned 8-bit integer	8

19643 **13.3.4.4.1 IEEE Address Field**

19644 The *IEEE address* field is 64-bits in length and specifies the unique IEEE identifier for each single node.

19645 **13.3.4.4.2 Endpoint Identifier Field**

19646 The *endpoint identifier* field is 8-bits in length and specifies the identifier of the endpoint on which the sub-device is  
 19647 implemented. This value is determined by the application and can be freely chosen by the application in the range  
 19648 0x01 – 0xf0.

19649 **13.3.4.4.3 Profile Identifier Field**

19650 The *profile identifier* field is 16-bits in length and specifies the identifier of the profile supported by the sub-device.  
 19651 This value shall correspond to the *application profile identifier* field of the simple descriptor.

19652 **13.3.4.4.4 Device Identifier Field**

19653 The *device identifier* field is 16-bits in length and specifies the identifier of the device description supported by the  
 19654 sub-device. This value shall correspond to the *application device identifier* field of the simple descriptor.

19655 **13.3.4.4.5 Device Version Field**

19656 The *device version* field is 4-bits in length and specifies the version of the device description supported by the sub-  
 19657 device. This value shall correspond to the *application device version* field of the simple descriptor.

19658 **13.3.4.4.6 Number of Group Identifiers Field**

19659 The *number of group identifiers* field is 8-bits in length and specifies the number of unique group identifiers required  
 19660 by the application on that specific endpoint.

19661 **13.3.4.4.7 Sort Tag Field**

19662 The *sort tag* field is 8-bits in length and specifies a sorting of the sub-devices, if required. A value of 0x00 indicates  
 19663 that the field is not sorted. Other values indicate the order in the list.

19664 **13.3.4.5 Inter-PAN frame format<sup>162</sup>**

19665 When using the inter-PAN frame format for touchlink commissioning, frames shall be either broadcast or  
 19666 unicast directly to the recipient, depending on the frame (i.e. indirect transmissions are not permitted).  
 19667 The general format of an inter-PAN frame is illustrated below.

19668

---

<sup>162</sup> CCB 2105

Group	Field name	Octets	Description
MAC header	Frame control	2	Frame Type = 0b001 Security Enabled = 0 Frame Pending = 0 ACK Request = 0 (no ACK requested) or 1 (ACK requested) Intra-PAN = 0 Dest. Addressing Mode = 0b10 (short address) or 0b11 (extended address) Frame Version = As appropriate Source Addressing Mode = 0b11
	Sequence number	1	As appropriate
	Destination PAN ID	2	0xffff
	Destination address	2/8	0xffff if broadcast IEEE address of destination otherwise
	Source PAN ID	2	As appropriate
	Source address	8	IEEE address of source
NWK header	Frame control	2	Frame type = 0b11 Protocol version = as appropriate Remaining sub-fields ≡ 0
APS header	Frame control	1	Frame type = 0b11 Delivery mode = 0b00 (unicast) or 0b10 (broadcast) ACK format = 0 Security = 0 ACK request = 0 Extended header present = 0
	Group address	0	Not included
	Cluster identifier	2	0x1000
	Profile identifier	2	0xc05e
ZCL header	Frame control	1	Frame type = 0b01 Manufacturer specific = As appropriate <sup>163</sup> Direction = 0 (client to server) or 1 (server to client) Disable default response = 1
	Manufacturer code	0/2	Included only if the manufacturer specific sub-field is set to 1.
	Transaction sequence number	1	Incremented for every transmission of a command
	Command identifier	1	See clause 13.3
ZCL payload	Command payload	Variable	See clause 13.3
MAC footer	Frame check sequence	2	As appropriate for the frame

Figure 13-34. General format of an inter-PAN frame



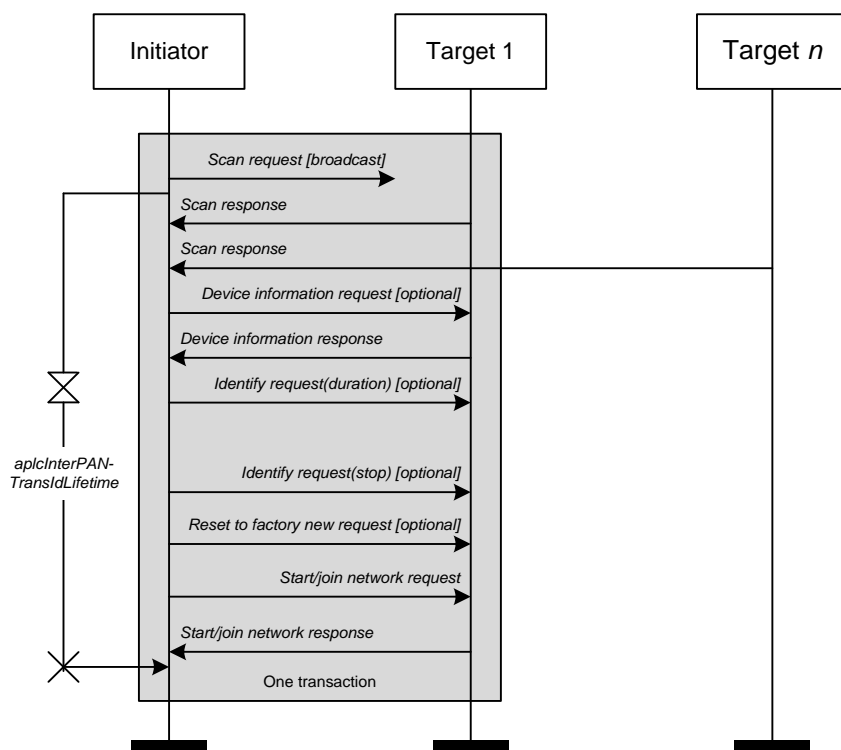
19670

19671 **13.3.4.6 Inter-PAN Transaction Identifier**

19672 All *touchlink commissioning* cluster inter-PAN command frames shall carry a 32-bit transaction identifier.

19673 The transaction identifier shall be created by the initiator of a *scan request* inter-PAN command frame and shall be  
 19674 random, non-zero and non-sequential. Related inter-PAN command frames which follow the *scan request*, i.e., *scan*  
 19675 *response*, *device information request/response*, *identify request*, *reset to factory new request*, *network start*  
 19676 *request/response*, *network join router request/response* and *network join end device request/response* define the scope  
 19677 of a transaction (illustrated in Figure 13-35) and shall carry the same transaction identifier as was defined in the *scan*  
 19678 *request*. While within the scope of a transaction (and for at most *aplInterPANTransIdLifetime*), the transaction  
 19679 identifier is said to be valid.

19680 **Figure 13-35. Scope of a touchlink commissioning inter-PAN transaction**



19681  
 19682 If a target, receiving a *scan request* inter-PAN command frame, is a sleeping end device, it shall enable its receiver  
 19683 while the transaction identifier is valid or for at most *aplInterPANTransIdLifetime* seconds after reception of the  
 19684 original *scan request* inter-PAN command frame. A device may disable its receiver before  
 19685 *aplInterPANTransIdLifetime* seconds have elapsed if the transaction has successfully completed and the device has  
 19686 started or joined the network.

19687 During a transaction, a device shall only accept inter-PAN command frames that contain a valid transaction identifier,  
 19688 i.e., inter-PAN command frames from within a transaction that have the same transaction identifier as was received in  
 19689 the *scan request* inter-PAN command frame, unless a device wants to start a new transaction after receiving a new  
 19690 *scan request* inter-PAN command frame from the same or another initiator carrying a new transaction identifier.

<sup>163</sup> When using the inter-PAN command frames defined in the Touchlink Commissioning cluster, the manufacturer specific sub-field of the ZCL header frame control field shall be set to 0.

### 19691 **13.3.4.7 Commissioning Scenarios**

19692 Touchlink commissioning between devices is performed from an *initiator* to a *target*, both of which can be  
19693 implemented from either an end device or a router. The commissioning mechanisms depend on whether the initiator  
19694 is factory new or non-factory new. If the initiator is factory new, it requests a new network to be started and if the  
19695 initiator is non-factory new it requests the target to join its network. If the target is non-factory new and already part  
19696 of a network, it can be *stolen* onto the network of the initiator. However, the target can decide whether to accept a  
19697 request to start a new network or join an existing network when requested to do so by the initiator. If the initiator is a  
19698 factory new end device, it must be commissioned with a router target so that a new network can be formed.

19699 For detailed information on touchlink commissioning, see the Base Device Behavior Specification.

### 19700 **13.3.4.8 Address Assignment**

19701 Network addresses and group identifiers are assigned by address assignment capable devices and all network addresses  
19702 and group identifiers must be unique.

#### 19703 **13.3.4.8.1 Network Address Assignment**

19704 Network addresses are assigned by devices that are address assignment capable. All network addresses must be unique.  
19705 The method used to ensure this is to assign subdivisions of the available address space to devices that join the network  
19706 and that are address assignment capable.

19707 Since ZigBee reserves the network address 0x0000 for the coordinator and the address range (0xfff8 ... 0xffff) for  
19708 broadcast, the total touchlink network address space is defined in the range (0x0001 ... 0xff7). Devices that are  
19709 address assignment capable shall keep track of their current free network address range, ( $N_{min}$  ...  $N_{max}$ ). When such  
19710 a device is factory-new,  $N_{min} = 0x0001$  and  $N_{max} = 0xff7$ .

19711 When a factory-new initiator device, which is address assignment capable, has just formed a new network, it shall  
19712 assign itself the network address  $N_{min}$  (i.e., 0x0001) and then increment  $N_{min}$ , i.e., the range changes to (0x0002 ...  
19713 0xff7).

19714 When a device is joined to an existing network, it shall be assigned the first (i.e.,  $N_{min}$ ) network address from the free  
19715 network address range of the initiator through which it is joining. The initiator that started the network shall then  
19716 increment  $N_{min}$ .

19717 If a device cannot be assigned a network address, it shall not be permitted to operate on the network.

19718 If a device that is address assignment capable joins the network, it shall also receive its own free network address  
19719 range ( $N'_{min}$  ...  $N'_{max}$ ). The initiator shall split its own free network address range at an implementation specified point  
19720 and the upper range (i.e., highest in value) shall be assigned to the new address assignment capable device.

19721 If after splitting the free network address range, the resulting two address ranges are smaller than an implementation  
19722 specific threshold, the new device shall not be joined to the network.

#### 19723 **13.3.4.8.2 Group Identifier Assignment**

19724 Group identifiers are used when addressing a subset of devices using broadcast mechanisms and they are typically  
19725 used by a controller application residing at a certain endpoint. The group identifiers need to be unique in the network  
19726 and their range is (0x0001 ... 0xfeff). Group identifier 0x0000 is used for the default group in the *ZCL scene* cluster.  
19727 Group identifiers (0xff00 ... 0xffff) shall be reserved.

19728 The number of group identifiers needed by an application residing on an endpoint is given in the device information  
19729 table. Since group identifier assignment is linked to network address assignment, the total number of group identifiers  
19730 needed by all endpoints on a node is reported in the *scan response* command frame. A device that is network address  
19731 assignment capable shall also be group identifier assignment capable and each shall keep track of their current free  
19732 group identifier range, ( $G_{min}$  ...  $G_{max}$ ). When such a device is factory-new,  $G_{min} = 0x0001$  and  $G_{max} = 0xfeff$ .

19733 When a factory-new initiator device which is assignment capable has just formed a new network, it shall take the  
19734 group identifiers, starting from  $G_{min}$  (i.e., 0x0001) for its own endpoints and shall then increment  $G_{min}$  with the number  
19735 of endpoints supported on the device.

19736 When a device is joined to the network, it shall receive a range of group identifiers for its endpoints and the initiator  
19737 shall then increment  $G_{min}$  with the number of endpoints supported on the new device.

19738 If a device that is about to be joined is also address assignment capable, it shall also receive a free group identifier  
19739 range ( $G'_{min} \cdots G'_{max}$ ), if possible. The initiator shall split its own free group identifier range at an implementation  
19740 specified point and the upper range (i.e., highest in value) shall be assigned to the new address assignment capable  
19741 device.

19742 If, after division of a free group identifier range, the resulting two group identifier ranges are smaller than an  
19743 implementation specific threshold, the new device shall not be joined to the network.

## 19744 13.3.4.9 Network Update

### 19745 13.3.4.9.1 Initiator Procedure

19746 If an initiator finds a device during device discovery that is part of the same network as the initiator but that reports a  
19747 network update identifier in its *scan response* inter-PAN command frame that is lower than that of the initiator, it may  
19748 generate and transmit a *network update request* inter-PAN command frame to the target using the unicast data service.

19749 The *network update request* inter-PAN command frame shall contain the current network parameters of the initiator  
19750 in the extended PAN identifier, network update identifier, logical channel and PAN identifier fields. In addition, the  
19751 *network update request* inter-PAN command frame shall also contain the network address of the target.

19752 Conversely, if an initiator finds a device during device discovery that is part of the same network as the initiator but  
19753 that reports a network update identifier in its *scan response* inter-PAN command frame that is higher than that of the  
19754 initiator, it shall update its stored network update identifier and logical channel with the values received in the *scan*  
19755 *response* inter-PAN command frame and change to the new channel accordingly.

19756 If the initiator is an end device, it shall then perform a network rejoin request by issuing the NLME-JOIN.request  
19757 primitive to the NWK layer, ensuring the *RejoinNetwork* parameter is set to indicate that the device is joining the  
19758 network using the NWK rejoining procedure. If the network rejoin was successful (indicated by the reception of the  
19759 NLME-JOIN.confirm), the initiator can use the network to communicate.

### 19760 13.3.4.9.2 Target Procedure

19761 On receipt of the *network update request* inter-PAN command frame with a valid transaction identifier (i.e.,  
19762 immediately following a device discovery) by a target, it shall first compare the values of the extended PAN identifier  
19763 and PAN identifier fields with its corresponding stored valued. If the two values are not identical, the target shall  
19764 discard the frame and perform no further processing. If the two values are identical, the target shall then compare the  
19765 value of the network update identifier field with its corresponding stored value. If the value in the frame is higher than  
19766 its stored value, the target shall update its stored network update identifier and logical channel with the values received  
19767 in the *network update request* inter-PAN command frame, according to the policy described in TBD. Otherwise, the  
19768 target shall discard the frame and perform no further processing.

19769 The target shall not send a response to a *network update request* inter-PAN command frame.

### 19770 **13.3.4.10 Frequency Agility**

19771 Touchlink supports a channel change mechanism in an application-defined way. When the channel change mechanism  
19772 is instigated, the device shall broadcast a Mgmt\_NWK\_Update\_req command frame with the *scan channels* field set  
19773 to indicate the channel on which to begin operating, the *scan duration* field set to 0xfe (channel change request) and  
19774 the *nwkUpdateId* field set to the value of the *nwkUpdateId* attribute of the transmitting device, incremented by one.  
19775 This command frame shall be broadcast to all devices for which *macRxOnWhenIdle* is equal to True (i.e., a network  
19776 address of 0xffffd).

19777 Routers receiving this Mgmt\_NWK\_Update\_req command frame shall update their NIB and execute their channel  
19778 change procedure. End devices shall rejoin using the NWK rejoining procedure.

19779 Routers that have missed the Mgmt\_NWK\_Update\_req command frame can be brought back into the network through  
19780 a touch-link procedure. For this reason, a device shall indicate the value of its *nwkUpdateId* attribute when it responds  
19781 to a scan request via a *scan response* command frame.

19782 If a touch-link initiator wants to bring a router back into the network (i.e., if the value of the *nwkUpdateId* indicated  
19783 in the scan response command frame is older than the value of the *nwkUpdateId* attribute of the scan initiator), it shall  
19784 send a unicast inter-PAN *network update request* command frame.

19785 If a touch-link initiator detects a router reporting a *nwkUpdateId* attribute that is newer than its own *nwkUpdateId*  
19786 attribute, it shall update its network settings (i.e., logical channel, PAN identifier and *nwkUpdateId*) accordingly based  
19787 on the values found in the *scan response* command frame sent by that router. If the touch-link initiator is an end device,  
19788 it shall execute a re-join procedure.

19789 Note: the *nwkUpdateId* attribute can take the value 0x00 – 0xff and may wrap around so care must be taken when  
19790 comparing for newness. For consistency, each device shall determine the *nwkUpdateId* to use (ID) using the following  
19791 algorithm:

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

19792

### 19793 **13.3.4.11 Security**

19794 Devices in a ZigBee PRO network shall use ZigBee network layer security. Each network shall have its own network  
19795 key. In touchlink, the network key shall be generated randomly by the initiator that starts the new network.

19796 In this clause concatenation of strings is represented by the “||” symbol.

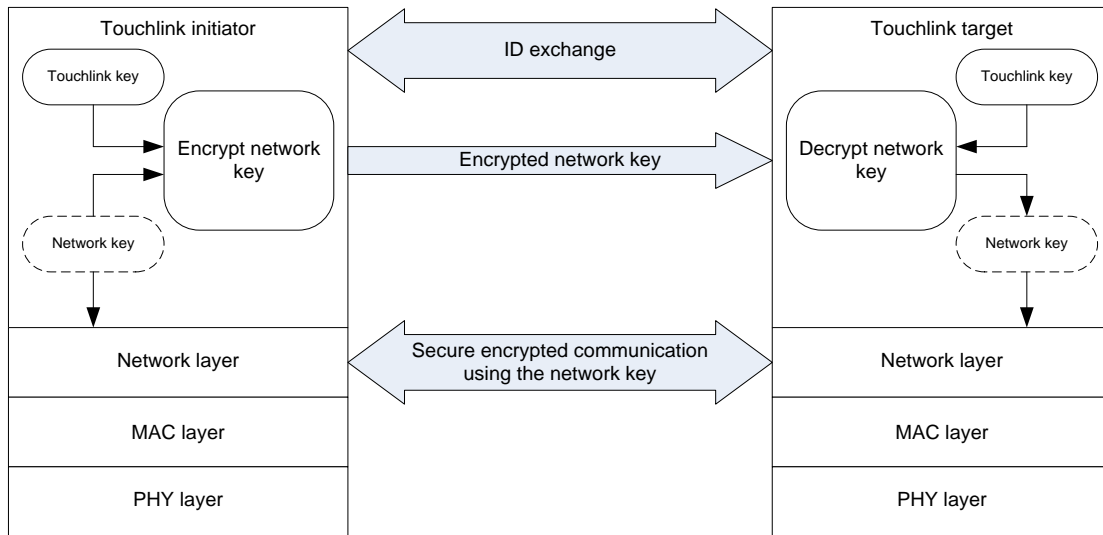
#### 19797 **13.3.4.11.1 Transferring the Network Key during Touchlink Commissioning**

19798 The touchlink security architecture is based on using a fixed secret key, known as the touchlink key, which shall be  
19799 stored in each device. During touchlink commissioning, all devices use the touchlink key to encrypt/decrypt the  
19800 exchanged network key.

19801 The architecture that is used to allow for a transfer the encrypted network key is depicted in Figure 13-36.

19802

**Figure 13-36. Overview of Touchlink Security**



19803

19804 In order to transfer the network key between the initiator and a possible target in a secure way, 16 possible algorithms  
 19805 can be used to encrypt the network key.

19806 The possible target shall indicate in the key bitmask field of its *scan response* inter-PAN command frame, transmitted  
 19807 during device discovery, which key encryption algorithms are supported.

19808 On receipt of each *scan response* inter-PAN command frame, the initiator shall compare the value in the received key  
 19809 bitmask field with its own stored key bitmask to find out if the two devices contain a common key. If no common key  
 19810 is found (i.e., the bitwise AND of the two is equal to zero), the initiator shall not select this target for further  
 19811 commissioning.

19812 If a common key is found (i.e., the bitwise AND of the two is not equal to zero), the initiator shall set the key index  
 19813 to the bit position corresponding to the matching key with the highest index, encrypts the network key using the  
 19814 appropriate algorithm, listed in Table 13-23, and includes both the index and the encrypted key it in the key index and  
 19815 encrypted network key fields, respectively, of the *network start request*, *network join router* or *network join end device*  
 19816 inter-PAN command frames.

19817

**Table 13-23. Key Encryption Algorithms**

Key index	Key description	Algorithm
0	Development key	See 13.3.4.11.4
1-3	Reserved	-
4	Master key	See 13.3.4.11.5
5-14	Reserved for future use	-
15	Certification key	See 13.3.4.11.5

19818 **13.3.4.11.2 Transferring the Network Key during Classical ZigBee**  
 19819 **Commissioning**

19820 During classical ZigBee commissioning where a device is being joined to a network without a trust center, a pre-  
 19821 installed link key is used to secure the transfer of the network key when authenticating. The pre-installed link key is a  
 19822 secret shared by all certified devices. It will be distributed only to certified manufacturers and is bound with a  
 19823 safekeeping contract.

19824 Prior to the successful completion of the certification, a certification pre-installed link key is used to allow testing.  
19825 The certification pre-installed link key shall have the value of:

```
Certification pre-installed      0xd0 0xd1 0xd2 0xd3 0xd4 0xd5 0xd6 0xd7
link key (0:15) =                0xd8 0xd9 0xda 0xdb 0xdc 0xdd 0xde 0xdf
```

19826 Additionally, if the decryption of the APS message fails with the key described above, devices shall try to decode the  
19827 APS message using the known default trust center link key.

### 19828 13.3.4.11.3 ZigBee Settings

19829 The following ZigBee security related NIB attributes shall be set (See [ZigBee], Section 4.3.3):

- 19830 • nwkSecurityLevel: 0x05 (use data encryption and frame integrity),
- 19831 • nwkAllFresh: False (do not check frame counter),
- 19832 • nwkSecureAllFrames: True (only accept secured frames).

### 19833 13.3.4.11.4 Key Index 0

19834 The network key encryption algorithm with a key index equal to 0 is known as the development key. This algorithm  
19835 encrypts the network key with AES in ECB mode in one single step where the AES key is equal to:

19836  $\text{"PhLi"} \parallel \text{TrID} \parallel \text{"CLSN"} \parallel \text{RsID}$

19837 Where TrID is the transaction identifier field of the original *scan request* command frame passed between the initiator  
19838 and target and RsID is the response identifier of the *scan response* command frame passed between the target and the  
19839 initiator (both values are random 32-bit integers). The ASCII characters in quotes ("") should be converted to their  
19840 equivalent hexadecimal byte values, with the leftmost character being the leftmost byte.

19841 For example:

<b>Encrypted Network Key (0:15)</b>	0x48 0x3c 0x2b 0x19 0x7c 0x27 0xc3 0xcc 0x76 0xa3 0xd6 0x3b 0x2e 0xa8 0xdb 0x0b
<b>Transaction identifier</b>	0xea9cd138
<b>Response identifier</b>	0x8f8dbab4
<b>Resulting AES Key (0:15)</b>	0x50 0x68 0x4c 0x69 0xea 0x9c 0xd1 0x38 0x43 0x4c 0x53 0x4e 0x8f 0x8d 0xba 0xb4
<b>Decrypted Network Key (0:15)</b>	0xac 0xbe 0xf1 0x44 0x70 0x27 0xd8 0xd9 0x5a 0xfa 0x42 0xb0 0x77 0xe4 0x88 0xa5

19842 Note: The development key (key index 0) shall only be used during the development phase of Light Link products.  
19843 Commercial Light Link products shall not use nor indicate having support for the development key.

### 19844 13.3.4.11.5 Key Index 4 and 15

#### 19845 13.3.4.11.5.1 Key Usage

19846 The touchlink security details described in this section apply to key index 4 and 15 of Table 13-23. The secure NWK  
19847 key transport methods indicted by key index 4 and 15 use the same algorithm, as described in section 13.3.4.11.5.2.  
19848 However, they differ in the type of key they use for NWK key protection.

#### 19849 13.3.4.11.5.1.1 Master Key (key index 4)

19850 The touchlink master key is a secret shared by all certified devices. It will be distributed only to certified manufacturers  
19851 and is bound with a safekeeping contract.

19852 The device using the touchlink master key in combination with the algorithm described in this document shall always  
19853 set bit 4 in the key bitmask field of the *scan response* command frame to 0b1 (see 13.3.2.3.1).

#### 19854 **13.3.4.11.5.1.2 Certification Key (key index 15)**

19855 Prior to the successful completion of the certification, a certification key is used to allow testing of the security  
19856 mechanisms as specified in this document. The certification key shall have the value of:

Certification key (0:15) =   0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7  
  0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf

19857 The device using the certification key in combination with the algorithm described in this document shall always set  
19858 bit 15 in the key bitmask field of the *scan response* command frame to 0b1 (see 13.3.2.3.1).

19859 The certification key may also be used during the development phase of products. However, commercial products  
19860 shall not use nor indicate having support for the certification key.

#### 19861 **13.3.4.11.5.2 Algorithm**

##### 19862 **13.3.4.11.5.2.1 Encrypting Network Keys for Touchlink Initiator**

19863 The touchlink initiator shall perform the following steps to encrypt the network key and transport it to the touchlink  
19864 target:

- 19865 • Exchange of transaction identifier and response identifier as part of the touchlink procedure.
- 19866 • Derive the ephemeral transport key (see Figure 13-37) from the transaction identifier, response identifier and  
19867 touchlink master or certification key, as described in 13.3.4.11.5.2.3.
- 19868 • Encrypt the network key using the calculated transport key and the AES ECB mode, as described in  
19869 13.3.4.11.5.2.3.
- 19870 • Transmit the encrypted network key to the touchlink target as part of the touchlink procedure.

##### 19871 **13.3.4.11.5.2.2 Decrypting network keys for touchlink target**

19872 The touchlink target shall perform the following steps to decrypt the network key received from the touchlink initiator:

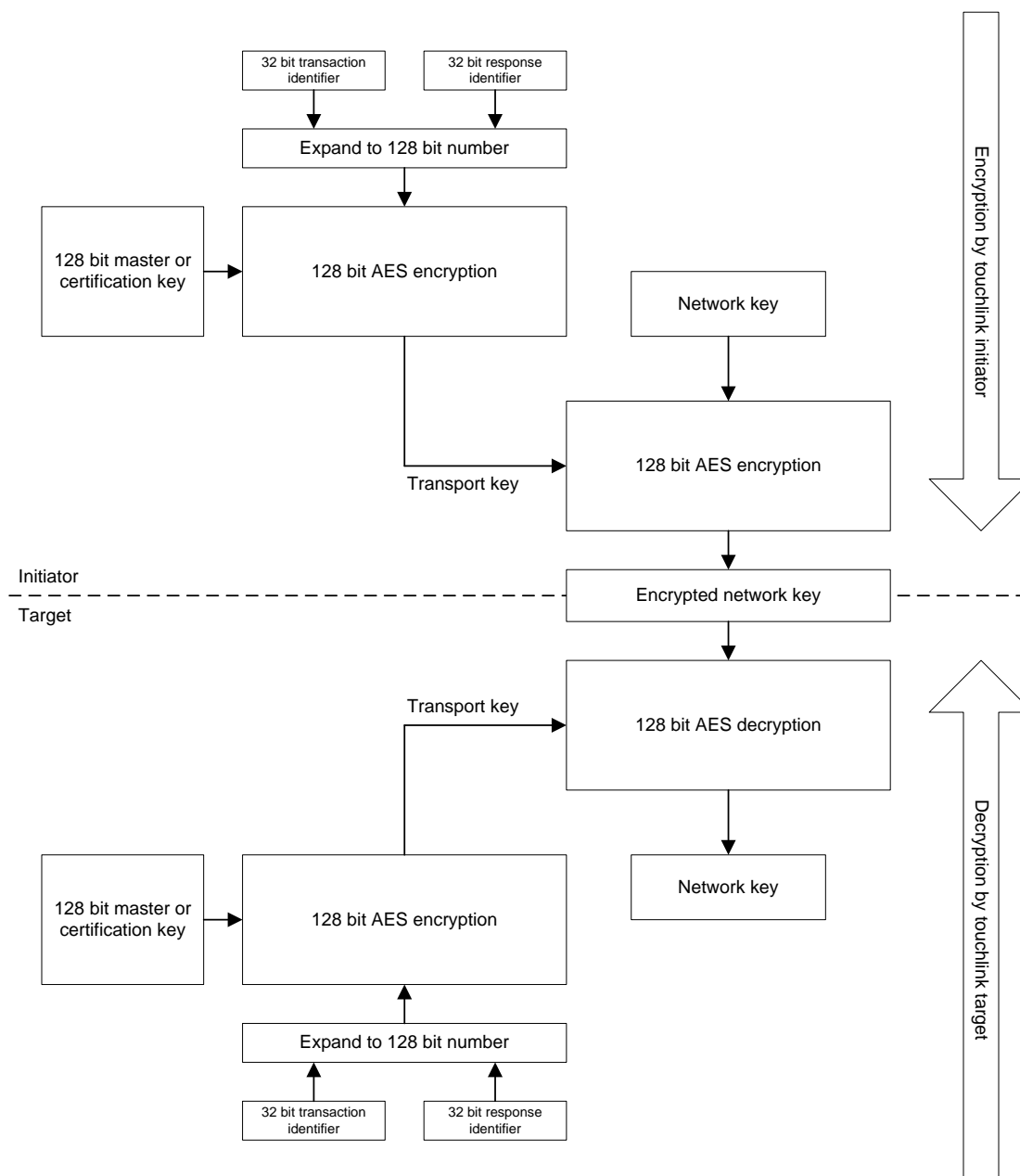
- 19873 • Exchange of transaction identifier and response identifier as part of the touchlink procedure.
- 19874 • Receive the encrypted network key as part of the touchlink procedure.
- 19875 • Derive the transport key (see Figure 13-37) from the transaction identifier, response identifier and touchlink  
19876 master or certification key, as described in 13.3.4.11.5.2.3.
- 19877 • Decrypt the received encrypted network key by using the calculated transport key and the AES ECB mode,  
19878 as described in 13.3.4.11.5.2.3.
- 19879 • Store the received network key in the NIB parameter of the touchlink target.

##### 19880 **13.3.4.11.5.2.3 Calculations Required for the Encryption/Decryption of the Network Key**

19881 The encryption/decryption key calculation to encrypt/decrypt the network key is illustrated in Figure 13-37.

19882

**Figure 13-37. Steps Required to Encrypt/Decrypt the Network Key**



19883

19884 Unless explicitly specified otherwise, all numbers in this chapter are formatted little Endian, i.e., with their least  
19885 significant octet first.

19886 The basic ingredients to perform the encryption/decryption of the network key are:

- 19887 • The 32 bit transaction identifier
- 19888 • The 32 bit response identifier
- 19889 • The touchlink master or certification key

19890 The encryption of the network key is performed by the following processing steps:

- 19891 8. Merge and expand the transaction identifier and response identifier into a 128 bit number by concatenating them  
19892 (in Little Endian representation) as follows:



- 19893 Transaction identifier || transaction identifier || response identifier || response identifier.
- 19894 9. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained  
19895 from step 1 as *plaintext*, and touchlink master or certification key as *key*.
- 19896 10. Encrypt the network key by executing the 128 bit AES encryption using the network key as *plaintext* and the  
19897 transport key obtained from step 2 as *key*.
- 19898 The decryption of the network key is performed by the following processing steps:
- 19899 11. Merge and expand the transaction identifier and response identifier into a 128 bit number, as described in step 1.
- 19900 12. Calculate the transport key by executing the 128 bit AES encryption with the expanded 128 bit number obtained  
19901 from step 4 used as *plaintext*, and touchlink master or certification key as *key*.
- 19902 13. Decrypt the network key by executing the 128 bit AES decryption with the transport key obtained from step 5  
19903 as *key* and the encrypted network key as *ciphertext*.
- 19904 All AES functions used in steps 2, 3, and 5 above shall use AES encryption in ECB mode and the AES function in  
19905 step 6 shall use AES decryption in ECB mode.

### 19906 13.3.4.11.6 Touchlink Security Test Vectors

19907 This annex provides sample test vectors for the touchlink security specification (as defined in sub-clause 13.3.4.11),  
19908 in order to assist in building interoperable security implementations.

#### 19909 13.3.4.11.6.1 Touchlink initiator operation

<b>Touchlink Certification Key (0:15)</b>	0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf
<b>Transaction ID</b>	0x3eaa2009
<b>Response ID</b>	0x88762fb1
<b>Expanded input (0:15)</b>	0x3e 0xaa 0x20 0x09 0x3e 0xaa 0x20 0x09 0x88 0x76 0x2f 0xb1 0x88 0x76 0x2f 0xb1

19910

19911 After AES ECB encryption:

<b>Transport Key (0:15)</b>	0x66 0x9e 0x08 0xe4 0x02 0x77 0xed 0x9a 0xb3 0x6b 0x25 0x80 0x45 0x6b 0x41 0x76
<b>NWK key (0:15)</b>	0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee 0xff 0x00

19912

19913 After AES ECB encryption:

<b>Encrypted Network Key (0:15)</b>	0x83 0x22 0x63 0x68 0x73 0xa7 0xbb 0x2a 0x18 0x9a 0x53 0x70 0x8c 0x60 0x7b 0xd0
-------------------------------------	--

#### 19914 13.3.4.11.6.2 Touchlink target operation

<b>Touchlink Certification Key (0:15)</b>	0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc6 0xc7 0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf
---	--

<b>Transaction ID</b>	0x3eaa2009
<b>Response ID</b>	0x88762fb1
<b>Expanded input (0:15)</b>	0x3e 0xaa 0x20 0x09 0x3e 0xaa 0x20 0x09 0x88 0x76 0x2f 0xb1 0x88 0x76 0x2f 0xb1

19915

19916 After AES ECB encryption:

<b>Transport Key (0:15)</b>	0x66 0x9e 0x08 0xe4 0x02 0x77 0xed 0x9a 0xb3 0x6b 0x25 0x80 0x45 0x6b 0x41 0x76
<b>Received encrypted NWK key (0:15)</b>	0x83 0x22 0x63 0x68 0x73 0xa7 0xbb 0x2a 0x18 0x9a 0x53 0x70 0x8c 0x60 0x7b 0xd0

19917

19918 After AES ECB decryption:

<b>NWK key (0:15)</b>	0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee 0xff 0x00
-----------------------	--

19919 Note: the first (i.e., leftmost on the page) byte of the encrypted network key is sent first in the associated encrypted  
19920 network key fields of the network start request, network join router request and network join end device request inter-  
19921 PAN command frames.

19922

# CHAPTER 14 RETAIL

19923  
 19924 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 19925 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 19926 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 19927 using [*Rn*] notation.

## 14.1 General Description

### 14.1.1 Introduction

19929  
 19930 The clusters specified in this chapter are for use typically in retail applications, but may be used in any application  
 19931 domain.

### 14.1.2 Cluster List

19932  
 19933 This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of  
 19934 clarification.

19935 The clusters specified in this chapter are listed in Table 14-1.

**Table 14-1. Clusters Specified in this Chapter**

ID	Cluster Name	Description
0x0617	Retail Tunnel Cluster	Interface for manufacturer specific information to be exchanged
0x0022	Mobile Device Configuration Cluster	Interface to manage mobile devices in a network
0x0023	Neighbor Cleaning Cluster	Interface to manage mobile devices in a network
0x0024	Nearest Gateway Cluster	Interface to enable communication of nearest gateway to devices

## 14.2 Retail Tunnel (MSP Tunnel)

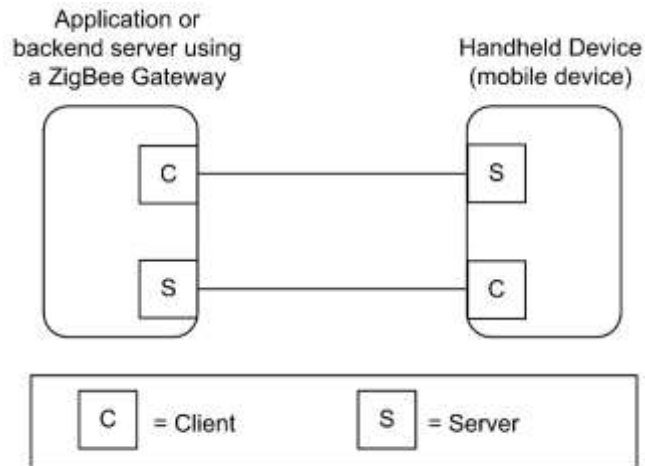
### 14.2.1 Overview

19937  
 19938  
 19939 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 19940 etc.

19941 This cluster provides an interface for transferring information encoded through a specific Manufacturer specific Profile  
 19942 from a device (e.g., a backend application using a gateway) to a handheld device (e.g., the Retail HHD). The messages  
 19943 that are transferred use a transfer APDU command as for other tunneling clusters defined (e.g., 11073 Protocol tunnel,  
 19944 or ISO 7818 tunnel).

19945

**Figure 14-1. Typical Usage of the Retail Tunnel Cluster**



*Note: Device names are examples for illustration purposes only*

19946

### 19947 **14.2.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 19948 **14.2.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	RTUN	Type 1 (client to server)

### 19949 **14.2.1.3 Cluster Identifiers**

Identifier	Name
0x0617	Retail Tunnel

## 19950 **14.2.2 Server**

### 19951 **14.2.2.1 Dependencies**

19952 This cluster may leverage on the Partition cluster in order to carry payloads not fitting into a single ZCL payload.

### 19953 **14.2.2.2 Attributes**

19954 The currently defined attributes for this cluster are listed in Table 14-2.

19955

**Table 14-2. Attributes of the Retail Tunnel cluster**

Id	Name	Type	Range	Acc	Def	M/O
0x0000	<i>ManufacturerCode</i>	uint16	0x1000 – 0x10ff	R	-	M
0x0001	<i>MSPProfile</i>	uint16	0xC000 – 0xFFFF	R	-	M

19956 **14.2.2.2.1 ManufacturerCode Attribute**

19957 The *ManufacturerCode* attribute specifies the manufacturer code relating the manufacturer of the device. This attribute  
 19958 can be used to match the proper protocol associated to the manufacturer of the device and tunneled through this cluster.  
 19959 See [Z12] Manufacturer Code Database.

19960 **14.2.2.2.2 MSPProfile Attribute**

19961 The *MSPProfile* attribute specifies the manufacturer specific profile used in the tunneled messages carried by the  
 19962 Transfer APDU commands. The *MSPProfile* attribute can be used to have the information of the proper protocol used  
 19963 by the communication entities supporting the MSP Tunnel cluster in order to properly decode the messages tunneled  
 19964 in this cluster.

19965 **14.2.2.3 Commands Received**

19966 Table 14-3 lists the cluster-specific commands that are received by the server.

19967 **Table 14-3. Cluster-specific Commands Received by the Server**

Command identifier field value	Description	Mandatory / Optional
0x00	Transfer APDU	M

19968 **14.2.2.3.1 Transfer APDU Command**

19969 **14.2.2.3.1.1 Payload Format**

19970 The Transfer APDU command shall be formatted as illustrated in Figure 14-2.

19971 **Figure 14-2. Format of the Transfer APDU Command**

<b>Bits</b>	<b>Variable</b>
<b>Data Type</b>	Octet String
<b>Field Name</b>	APDU

19972 **14.2.2.3.1.2 APDU Field**

19973 The APDU field is of variable length and is an APDU as defined in the *MSPProfile* attribute of the Manufacturer  
 19974 indicated by the *ManufacturerCode* attribute.

19975 **14.2.2.3.1.3 When Generated**

19976 This command is generated when a message has to be transferred across a MSP tunnel. The message can be only  
19977 decoded by the recipient entity if it is provided by the proper decodes of the Manufacturer specific profile as defined  
19978 in [Z7].

#### 19979 **14.2.2.3.1.4 Effect on Receipt**

19980 On receipt of this command, a device shall process the APDU according to the specific MSP transported.

#### 19981 **14.2.2.4 Commands Generated**

19982 No cluster-specific commands are generated by the server cluster.

### 19983 **14.2.3 Client**

19984 The client has no dependencies, no cluster specific attributes. The client does not receive any cluster-specific  
19985 commands. The client generates the cluster-specific commands detailed in 14.2.2.3.

## 19986 **14.3 Mobile Device Configuration**

---

### 19987 **14.3.1 Overview**

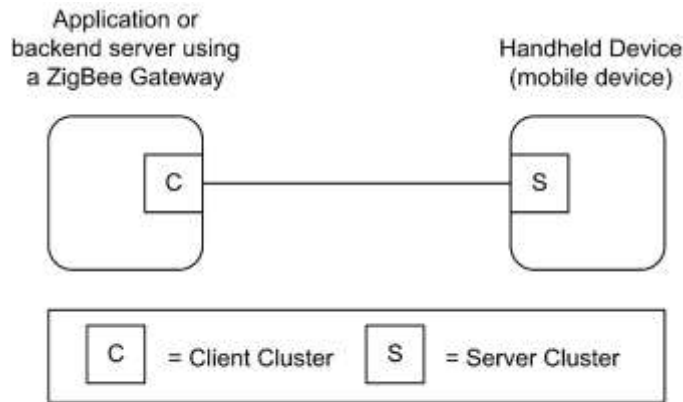
19988 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
19989 etc.

19990 This cluster provides an interface to enable the management of mobile devices in a network.

19991 If a stack supports neighbor entry aging, the mobile device will be able to use this cluster to refresh the information in  
19992 the parent/neighbor. An application will be also able to configure aging timeout (using the Neighbor cleaning cluster)  
19993 greater than *KeepAliveTime*, managing in this way the timeout used for cleaning neighbor table setting appropriate  
19994 value. Besides, *Rejoin timeout* can be used to allow the device force a rejoin and then allow the mobile device solution  
19995 to work with stacks not supporting the cleaning of the neighbor tables.

19996

**Figure 14-3. Typical Usage of the Mobile Device Configuration Cluster**



*Note: Device names are examples for illustration purposes only*

19997

19998 **14.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

19999 **14.3.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	MOBCFG

20000 **14.3.1.3 Cluster Identifiers**

Identifier	Name
0x0022	Mobile Device Configuration

20001 **14.3.2 Server**

20002 **14.3.2.1 Dependencies**

20003 This cluster should be supported by devices that are mobile in the network. The devices building the network  
 20004 infrastructure should use the Neighbor Cleaning Cluster to manage the loss of the mobile devices from the radio range.

20005 **14.3.2.2 Attributes**

20006 The currently defined attributes for this cluster are listed in Table 14-4.

20007

Table 14-4. Attributes of the Mobile Device Cleaning Cluster

Identifier	Name	Type	Range	Acc	Unit	Default	M/O
0x0000	<i>KeepAliveTime</i>	uint16	0x0001- 0xFFFF	RW	Seconds	15 seconds (0x000F)	M
0x0001	<i>RejoinTimeout</i>	uint16	0x0000- 0xFFFF	RW	Seconds	0xFFFF (Never)	M

20008 **14.3.2.2.1 KeepAliveTime Attribute**

20009 The *KeepAliveTime* attribute specifies the time period to elapse before a mobile device send a Keep Alive Notification  
20010 message to the manager of the network (e.g. application backend servers using a gateway). Please note that a value of  
20011 this attribute equal to 0xFFFF means that the mobile device shall not send *KeepAliveNotification* messages. This  
20012 attribute is used to “refresh” neighbor table information on its parent devices, avoiding expiration or aging of the  
20013 correspondent entry.

20014 **14.3.2.2.2 RejoinTimeout Attribute**

20015 The *RejoinTimeout* attribute specifies the time after which the device shall perform a secure network rejoin to clean  
20016 the entries in the neighbor table for parent devices not cleaning them with the Neighbor Cleaning Cluster. Please note  
20017 that a value of this attribute equal to 0xFFFF means that the mobile device is not requested to perform the network  
20018 Rejoin to clean the mesh. (Note: The mobile device may choose to transmit a Network Leave frame to the short address  
20019 being cleaned.)

20020 **14.3.2.3 Commands Received**

20021 No cluster-specific commands are received by the server side of this cluster.

20022 **14.3.2.4 Commands Generated**

20023 Table 14-5 lists cluster-specific commands that are generated by the server.

20024 Table 14-5. Cluster-specific Commands Generated by the Server

Command Id	Description	M/O
0x00	<i>Keep Alive Notification</i>	M

20025 **14.3.2.4.1 Keep Alive Notification Command**20026 **14.3.2.4.1.1 Payload Format**

20027 The Keep Alive Notification command shall be formatted as illustrated in Figure 14-4.

20028 Figure 14-4. Format of the Keep Alive Notification Command

Bits	Variable	Variable
Data Type	uint16	uint16
Field Name	<i>KeepAliveTime</i>	<i>RejoinTimeout</i>



20029 **14.3.2.4.1.1.1 KeepAliveTime Field**

20030 This field corresponds to the *KeepAliveTime* attribute.

20031 **14.3.2.4.1.1.2 RejoinTimeout Field**

20032 This field corresponds to the *RejoinTimeout* attribute.

20033 **14.3.2.4.1.2 When Generated**

20034 This command is generated when a time greater than *KeepAliveTime* attribute elapses.

20035 **14.3.2.4.1.3 Effect on Receipt**

20036 On receipt of this command, a parent or neighbor device shall refresh neighbor table information on the mobile node  
20037 sending the Keep Alive Notification by resetting the timers managing the expiration of the entries in the neighbors  
20038 table.

20039 **14.3.3 Client**

20040 The client has no dependencies, no cluster specific attributes. The client receives the commands specified in section  
20041 14.2.2.4. The client does not generate any cluster-specific commands.

20042 **14.4 Neighbor Cleaning**

---

20043 **14.4.1 Overview**

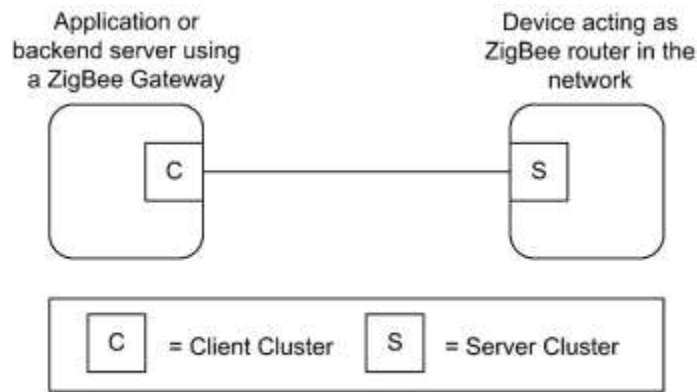
20044 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
20045 etc.

20046 This cluster provides an interface to enable the management of mobile devices in a network.

20047 If a stack supports neighbor entry aging, the mobile device will be able to use this cluster to clean the information in  
20048 the parent/neighbor. An application will be able to configure the aging timeout greater than a *KeepAliveTime* (attribute  
20049 supported by a mobile device), managing in this way the timeout used for cleaning neighbor table setting appropriate  
20050 value.

20051

**Figure 14-5. Typical Usage of the Neighbor Cleaning Cluster**



*Note: Device names are examples for illustration purposes only*

20052

### 20053 14.4.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 20054 14.4.1.2 Classification

Hierarchy	Role	PICS Code
Base	Utility	NBCLEAN

### 20055 14.4.1.3 Cluster Identifiers

Identifier	Name
0x0023	Neighbor Cleaning

## 20056 14.4.2 Server

### 20057 14.4.2.1 Dependencies

20058 This cluster should be supported by devices that are acting as routers for Mobile devices in the network; besides, the  
 20059 mobile devices within the network infrastructure (e.g., Hand Held devices or Mobile phones) should use the Mobile  
 20060 Device Configuration Cluster to take advantage of the mobility feature.

### 20061 14.4.2.2 Attributes

20062 The currently defined attributes for this cluster are listed in the following table.

20063

**Table 14-6. Attributes of the Neighbor Cleaning Cluster**

Id	Name	Type	Range	Acc	Unit	Default	M/O
0x0000	<i>NeighborCleaningTimeout</i>	uint16	0x0001 - 0xFFFF	RW	Seconds	30 seconds (0x001E)	M

20064 **14.4.2.2.1 NeighborCleaningTimeout Attribute**

20065 The *NeighborCleaningTimeout* attribute specifies the time period to elapse without receiving any messages from a  
 20066 neighbor device (router or end device) which is a mobile device, before cleaning its neighbor table entry. (Note: The  
 20067 cleaning device may choose to transmit a Network Leave frame to the short address being cleaned.)

20068 **14.4.2.3 Commands Received**

20069 Table 14-7 lists cluster-specific commands which are received by the server side of this cluster.

20070 **Table 14-7. Cluster-specific Commands Generated by the Server**

Command Id	Description	M/O
0x00	<i>PurgeEntries</i>	M

20071 **14.4.2.3.1 PurgeEntries Command**

20072 **14.4.2.3.1.1 Payload Format**

20073 The *PurgeEntries* command has no payload.

20074 **14.4.2.3.1.2 When Generated**

20075 This command is generated by the manager of the network supporting the mobile devices in order to force the cleaning  
 20076 of the neighbor table entries.

20077 **14.4.2.3.1.3 Effect on Receipt**

20078 On receipt of this command, a parent or neighbor device should clean the neighbor tables to delete aged entries; please  
 20079 notice that this feature can be executed only if enabled by the stack.

20080 **14.4.2.4 Commands Generated**

20081 No cluster-specific commands are generated by the server.

20082 **14.4.3 Client**

20083 The client has no dependencies and no cluster specific attributes. The client does not receive any cluster-specific  
 20084 commands. The client does generate the cluster-specific commands specified in 14.4.2.3.

20085 **14.5 Nearest Gateway**

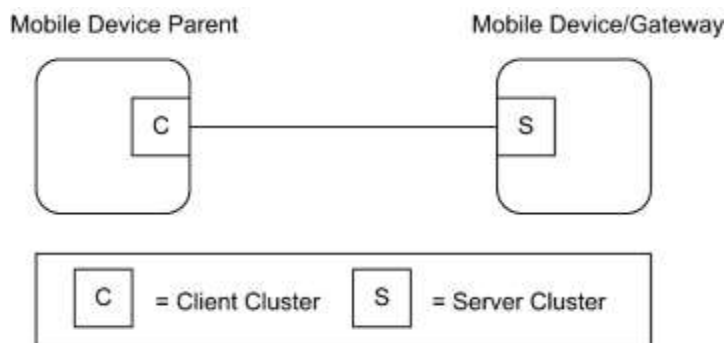
20086 **14.5.1 Overview**

20087 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
20088 etc.

20089 This cluster provides an interface to enable the dissemination of “nearest gateway” information.

20090 Based on MTORR information initiated by gateway devices (concentrator), the remaining routers in the network can  
20091 determine which gateway is closest based on path cost, i.e., the “nearest gateway.” The cluster allows that information  
20092 to be communicated to devices in the network that need that information.

20093 **Figure 14-6. Typical Usage of the Nearest Gateway Cluster**



Note: Device names are examples for illustration purposes only

20094

20095 **14.5.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

20096 **14.5.1.2 Classification**

Hierarchy	Role	PICS Code
Base	Utility	NEARGW

20097 **14.5.1.3 Cluster Identifiers**

Identifier	Name
0x0024	Nearest Gateway

20098 **14.5.2 Server**

20099 **14.5.2.1 Dependencies**

20100 This cluster should be supported by devices that are mobile in the network and, optionally, gateway devices.

20101 **14.5.2.2 Attributes**

20102 The currently defined attributes for this cluster are listed in the following table.

20103 **Table 14-8. Attributes of the Nearest Gateway Cluster**

Id	Name	Type	Range	Acc	Default	M/O
0x0000	<i>Nearest Gateway</i>	16-bit NWK address	0x0000- 0xFFFF8	RW	0x0000	M
0x0001	<i>New Mobile Node</i>	16-bit NWK address	0x0000- 0xFFFF8	W	0x0000	M

20104 **14.5.2.2.1 Nearest Gateway Attribute**

20105 The *Nearest Gateway* attribute specifies the gateway that is nearest in terms of path cost.

20106 **14.5.2.2.2 New Mobile Node Attribute**

20107 The *New Mobile Node* attribute specifies the new mobile node that joined the server.

20108 **14.5.2.3 Commands Received**

20109 No cluster-specific commands are received by the server side of this cluster.

20110 **14.5.2.4 Commands Generated**

20111 No cluster-specific commands are generated by the server side of this cluster.

20112 **14.5.3 Client**

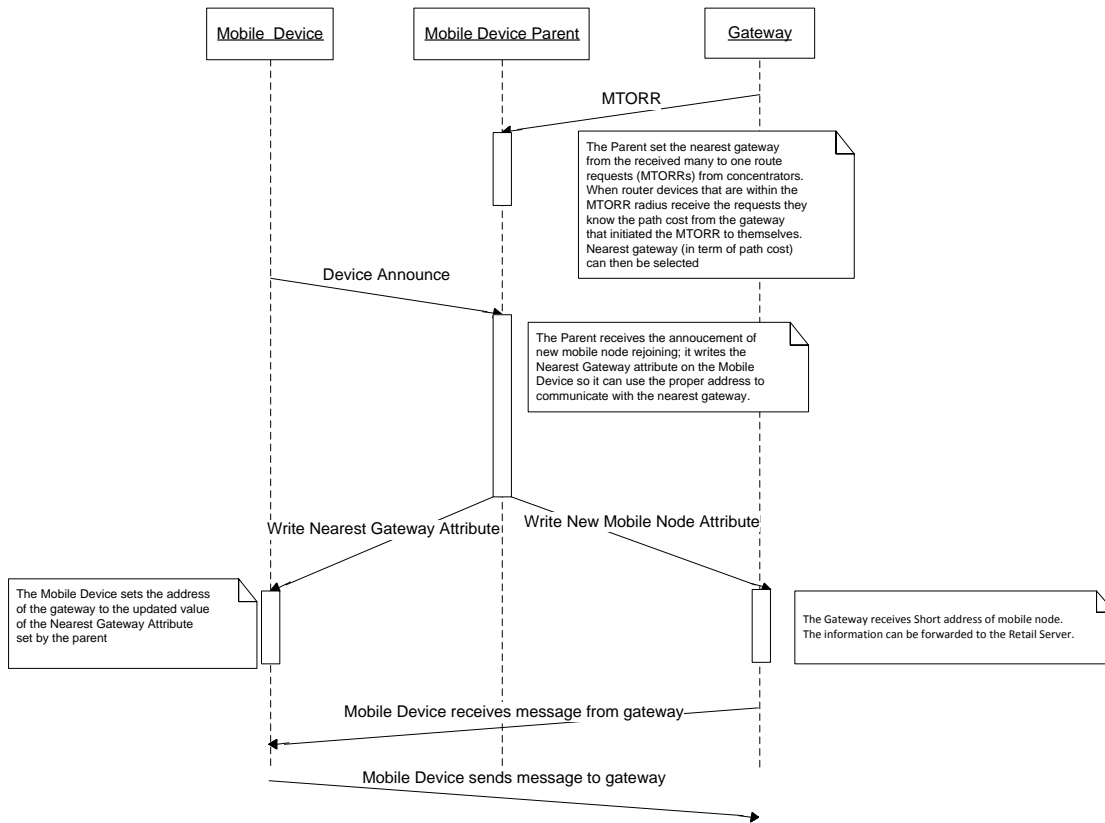
20113 The client has no dependencies and no cluster specific attributes. The client does not receive nor generate any cluster-specific commands.  
 20114

20115 **14.5.4 Examples of Use**

20116 Figure 14-7 describes an example of the possible use of the nearest gateway cluster.

20117

Figure 14-7. Sequence Diagram



20118

20119

## CHAPTER 15 APPLIANCE

20120

20121 The Cluster Library is made of individual chapters such as this one. See Document Control in the Cluster Library for  
 20122 a list of all chapters and documents. References between chapters are made using a *X.Y* notation where *X* is the chapter  
 20123 and *Y* is the sub-section within that chapter. References to external documents are contained in Chapter 1 and are made  
 20124 using [*Rn*] notation.

### 15.1 General Description

20125

#### 15.1.1 Introduction

20126

20127 The clusters specified in this chapter are for use typically in appliance management, but MAY be used in any  
 20128 application domain.

#### 15.1.2 Cluster List

20129

20130 This section lists the clusters specified in this chapter and gives examples of typical usage for the purpose of  
 20131 clarification.

20132 The clusters specified in this chapter are listed in Table 15-1.

20133

**Table 15-1. Appliance Management Clusters**

Id	Cluster Name	Description
0x001b	EN50523 Appliance Control	Commands and attributes for controlling household appliances
0x0b00	EN50523 Appliance Identification	Commands and attributes for appliance information and device settings
0x0b02	EN50523 Appliance Events and Alerts	Commands and attributes for appliance events and alerts
0x0b03	EN50523 Appliance Statistics	Commands and attributes for appliance statistics

### 15.2 EN50523 Appliance Control

20134

20135 This section describes the EN50523 Appliance Control cluster.

#### 15.2.1 Overview

20136

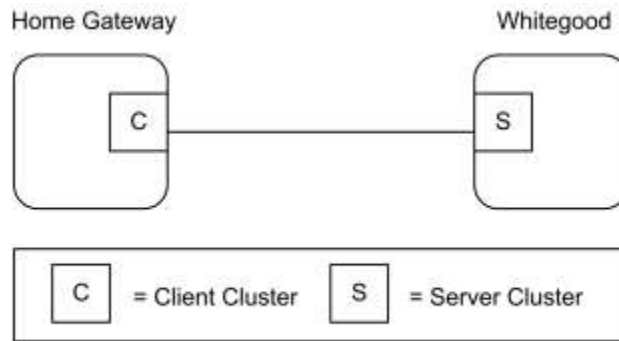
20137 Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification,  
 20138 identification, etc

20139 This cluster provides an interface to remotely control and to program household appliances. Example of control is  
 20140 Start, Stop and Pause commands.

20141 The status “read” and “set” is compliant to the EN50523 “Signal State” and “Execute Command” functional blocks.  
 20142 Appliances parameters (e.g., Duration and Remaining Time) have been added, since they were missing from the  
 20143 original specs.

20144

**Figure 15-1. Typical Usage of the Appliance Control Cluster**



*Note: Device names are examples for illustration purposes only*

20145

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

### 20149 15.2.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 20150 15.2.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APLNC	Type 2 (server to client)

### 20151 15.2.1.3 Cluster Identifiers

Identifier	Name
0x001b	EN50523 Appliance Control

## 20152 15.2.2 General Description

### 20153 15.2.3 Server Attributes

20154 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 20155 contain up to 256 attributes. Attribute identifiers are encoded such that the most significant byte specifies the attribute  
 20156 set and the least significant byte specifies the attribute within the set. The currently defined attribute sets are listed in  
 20157 Table 15-2.

20158

**Table 15-2. Appliance Control Attribute Set**

Attribute Set Identifier	Description
0x00	Appliance Functions



20159 **15.2.3.1 Appliance Functions Attribute Set**

20160 The Appliance Functions attribute set contains the attributes summarized in Table 15-3.

20161 These attributes control the Appliance cycle parameters. Each of them, as described below, corresponds to an  
 20162 Appliance internal status configuration.

20163 **Table 15-3. Attributes of the Appliance Functions Attribute Set**

Id	Name	Type	Range	Access	Default	M/O
0x0000	<i>StartTime</i>	uint16	0x0000 – 0xffff	RP	0x0000	M
0x0001	<i>FinishTime</i>	uint16	0x0000 – 0xffff	RP	0x0000	M
0x0002	<i>RemainingTime</i>	uint16	0x0000 – 0xffff	RP	0x0000	O

20164 **15.2.3.2 StartTime Attribute**

20165 *StartTime* attribute determines the time (either relative or absolute) of the start of the machine activity. Default format  
 20166 for Oven devices is absolute time. The default format for other appliances is relative time. *StartTime* SHOULD be set  
 20167 less than *FinishTime*.

20168 Table 15-4 provides details about time encoding which is used for *StartTime* attribute organization.

20169 **Table 15-4. Time Encoding**

Bit Range	Function	
0..5	Minutes ranging from 0 to 59	
6..7	Time encoding	
	Value	Enumeration
	0x0 0x1 0x2..0x3	RELATIVE ABSOLUTE Reserved
8..15	Hours ranging from 0 to 255 if RELATIVE encoding is selected 0 to 23 if ABSOLUTE encoding is selected	

20170 **15.2.3.3 FinishTime Attribute**

20171 *FinishTime* attribute determines the time (either relative or absolute) of the expected end of the machine activity.  
 20172 Default format for Oven is absolute time. The default format for other appliances is relative time. *FinishTime*  
 20173 SHOULD be set greater than *StartTime*.

20174 *FinishTime* attribute exploits time encoding reported in Table 15-4.

20175 **15.2.3.4 RemainingTime Attribute**

20176 *RemainingTime* attribute determines the time, in relative format, of the remaining time of the machine cycle. It  
 20177 represents the time remaining to complete the machine cycle and it is updated only during the RUNNING state of the  
 20178 Appliance. During the other states of the Appliance *RemainingTime* attribute is indicated as the not valid value “0”.

20179 *RemainingTime* attribute exploits time encoding reported in Table 15-4.

20180 **15.2.4 Server Commands Received**

20181 The command IDs for the Appliance Control cluster are listed in Table 15-5.

20182 **Table 15-5. Cluster-specific Commands Received by the Server**

Command Identifier Field Value	Description	M/O
0x00	Execution of a Command	O
0x01	Signal State	M
0x02	Write Functions	O
0x03	Overload Pause Resume	O
0x04	Overload Pause	O
0x05	Overload Warning	O

20183 **15.2.4.1 Execution of a Command**

20184 This basic message is used to remotely control and to program household appliances. Examples of control are START,  
 20185 STOP and PAUSE.

20186 **15.2.4.1.1 Payload Format**

20187 The Execution of a Command payload SHALL be formatted as illustrated in Figure 15-2.

20188 **Figure 15-2. Format of the Execution of a Command Payload**

<b>Octets</b>	1
<b>Data Type</b>	enum8
<b>Field Name</b>	Command Identification

20189 **15.2.4.1.1.1 Payload Details**

20190 The *Command Identification* field: the command identification is an 8-bits in length field identifying the command to  
 20191 be executed. The enumeration used for this field SHALL match Table 15-6.

20192 **Table 15-6. Command Identification Values**

Enumeration	Value	Description
START	0x01	Start appliance cycle

Enumeration	Value	Description
STOP	0x02	Stop appliance cycle
PAUSE	0x03	Pause appliance cycle
START SUPERFREEZING	0x04	Start superfreezing cycle
STOP SUPERFREEZING	0x05	Stop superfreezing cycle
START SUPERCOOLING	0x06	Start supercooling cycle
STOP SUPERCOOLING	0x07	Stop supercooling cycle
DISABLE GAS	0x08	Disable gas
ENABLE GAS	0x09	Enable gas
<i>Manufacturer Specific</i>	0x80..0xff	Manufacturer Specific

20193 **15.2.4.1.2 Effects on Receipt**

20194 On receipt of this command, the appliance SHALL execute the command given in the Command Identification field.  
 20195 The device application SHALL be informed of the imposed command (and potential personalized tasks could start,  
 20196 e.g., by means of a message to appliance Main Board controller).

20197 After the command execution, the appliance SHALL generate a Signal State Notification with the new appliance state.

20198 **15.2.4.2 Signal State Command**

20199 This basic message is used to retrieve Household Appliances status. This command does not have a payload.

20200 **15.2.4.2.1 Effects on Receipt**

20201 On receipt of this command, the device SHALL generate a Signal State Response command.

20202 **15.2.4.3 Write Functions Command**

20203 This basic message is used to set appliance functions, i.e., information regarding the execution of an appliance cycle.  
 20204 Condition parameters such as start time or finish time information could be provided through this command. A  
 20205 function is mirrored by the cluster attribute that represents its current state. See Effect on Receipt below to understand  
 20206 the difference between writing a function and writing an attribute value.

20207 **15.2.4.3.1 Payload Format**

20208 The Write Functions command frame SHALL be formatted as illustrated in Format of the Write Functions Command  
 20209 Frame.

20210

**Figure 15-3. Format of the Write Functions Command Frame**

<b>Octets</b>	Variable
<b>Field Name</b>	Write Functions record

20211

20212 Write Functions record SHALL be formatted as illustrated in Figure 15-4.

20213

**Figure 15-4. Format of the Write Functions Record Field**

<b>Octets</b>	2	1	Variable
<b>Data Type</b>	uint16	enum8	Variable
<b>Field Name</b>	Function identifier (i.e., attribute identifier)	Function data type	Function data

20214 **15.2.4.3.2 Payload Details**20215 The **Function identifier** field: the Function Identifier is 16-bits in length and SHALL contain the identifier of the  
20216 function that is to be written.20217 The **Function data type** field: the function data type field SHALL contain the data type identifier of the attribute that  
20218 is to be written.20219 The **Function data** field: the function data field is variable in length and SHALL contain the actual value of the  
20220 function that is to be written.20221 **15.2.4.3.3 Effects on Receipt**20222 On receipt of this command, the appliance SHALL set the function given in the Function identifier field. The Function  
20223 attribute is actually changed only when the appliance internal functions have been changed.20224 If attribute reporting is configured on some function attributes, an attribute reporting command is generated when the  
20225 attribute, and therefore internal appliance function is actually modified. In case attribute reporting is not used, the  
20226 correct execution of the Write Function command SHOULD be verified by using Read Attribute command to poll  
20227 the written attribute.20228 **15.2.4.4 Overload Pause Resume Command**20229 This command SHALL be used to resume the normal behavior of a household appliance being in pause mode after  
20230 receiving a Overload Pause command.20231 **15.2.4.4.1 Payload Format**

20232 The Overload Pause Resume Command SHALL have no payload.

20233 **15.2.4.4.2 Effects on Receipt**

20234 On receipt of this command, the appliance SHALL resume its operations.

20235 **15.2.4.5 Overload Pause Command**

20236 This command SHALL be used to pause the household appliance as a consequence of an imminent overload event.

20237 **15.2.4.5.1 Payload Format**

20238 The Overload Pause Command SHALL have no payload.

20239 **15.2.4.5.2 Effects on Receipt**

20240 On receipt of this command, the appliance SHALL pause its operations. In order to resume the normal operation an  
 20241 Overload Pause Resume command SHOULD be issued by the device supporting the client side of the Appliance  
 20242 control cluster.

20243 **15.2.4.6 Overload Warning Command**

20244 This basic message is used to send warnings the household appliance as a consequence of a possible overload event,  
 20245 or the notification of the end of the warning state.

20246 **15.2.4.6.1 Payload Format**

20247 The Overload Warning Command payload SHALL be formatted as illustrated in Figure 15-5.

20248 **Figure 15-5. Format of the Overload Warning Payload**

<b>Octets</b>	2
<b>Data Type</b>	enum8
<b>Field Name</b>	Warning Event

20249 **15.2.4.6.2 Payload Details**

20250 The Warning Event field represents the identifier of the events that needs to be communicated to the devices to alert  
 20251 about possible overload, as shown in Table 15-7.

20252 **Table 15-7. Format of the Event ID Enumerator**

<b>Event ID</b>	<b>Description</b>
0x00	Warning 1: overall power above “available power” level
0x01	Warning 2: overall power above “power threshold” level
0x02	Warning 3: overall power back below the “available power” level
0x03	Warning 4: overall power back below the “power threshold” level
0x04	Warning 5: overall power will be potentially above “available power” level if the appliance starts

20253 **15.2.4.6.3 Effects on Receipt**

20254 On receipt of this command, the appliance SHALL show the possible warning state on a display (e.g., showing an  
20255 icon with possible overload condition when activating the appliance in case of Warnings 1-2) or resume the normal  
20256 state in case of events showing the return on normal state (e.g., Warning 3-4).

20257 **15.2.5 Server Commands Generated**

20258 Table 15-8 lists commands that are generated by the server.

20259 **Table 15-8. Cluster-specific Commands Sent by the Server**

Command Identifier Field Value	Description	M/O
0x00	Signal State Response	M
0x01	Signal State Notification	M

20260 **15.2.5.1 Signal State Response Command**

20261 This command SHALL be used to return household appliance status, according to Appliance Status Values and  
20262 Remote Enable Flags Values.

20263 **15.2.5.1.1 Payload Format**

20264 The Signal State Response Command payload SHALL be formatted as illustrated in Figure 15-6.

20265 The Appliance Status field: the data field is an 8 bits in length enumerator identifying the appliance status. The  
20266 enumeration used for this field SHALL match the specifications in Table 15-9.

20267 The Remote Enable Flags and Device Status 2 field: the data field is an 8 bits in length unsigned integer defining  
20268 remote enable flags and potential appliance status 2 format. The unsigned integer used for this field SHALL match  
20269 the specifications in Table 15-10.

20270 The Appliance Status 2 field: the command identification is a 24 bits in length unsigned integer representing potential  
20271 non-standardized or proprietary data.

20272 **Figure 15-6. Format of the Signal State Response Command Payload**

<b>Octets</b>	1	1	0/3
<b>Data Type</b>	enum8	uint8	uint24
<b>Field Name</b>	Appliance Status	Remote Enable Flags and Device Status 2	Appliance Status 2

20273 **15.2.5.1.1.1 Payload Details**20274 **ApplianceStatus**

20275 *ApplianceStatus* represents the current status of household appliance. *ApplianceStatus* must be included as part of the  
20276 minimum data set to be provided by the household appliance device. *ApplianceStatus* is updated continuously as  
20277 appliance state changes.

20278 Table 15-9 provides states defined.

20279

**Table 15-9. Appliance Status Values**

Enumeration	Value	Description
OFF	0x01	Appliance in off state
STAND-BY	0x02	Appliance in stand-by
PROGRAMMED	0x03	Appliance already programmed
PROGRAMMED WAITING TO START	0x04	Appliance already programmed and ready to start (e.g., has not reached <i>StartTime</i> )
RUNNING	0x05	Appliance is running
PAUSE	0x06	Appliance is in pause
END PROGRAMMED	0x07	Appliance end programmed tasks
FAILURE	0x08	Appliance is in a failure state
PROGRAMME INTERRUPTED	0x09	The appliance programmed tasks have been interrupted
IDLE	0x0a	Appliance in idle state
RINSE HOLD	0x0b	Appliance rinse hold
SERVICE	0x0c	Appliance in service state
SUPERFREEZING	0x0d	Appliance in superfreezing state
SUPERCOOLING	0x0e	Appliance in supercooling state
SUPERHEATING	0x0f	Appliance in superheating state
<i>Manufacturer Specific</i>	0x80..0xff	Manufacturer specific value range

20280

20281 **RemoteEnableFlags Field**

20282 *RemoteEnableFlags* represents the current status of household appliance correlated with remote control.

20283 *RemoteEnableFlags* is mandatory and must be included as part of the minimum data set to be provided by the  
 20284 household appliance device.

20285 *RemoteEnableFlags* is updated continuously when appliance state remote-controllability changes.

20286 Table 15-10 provides details about flags organization.

20287

**Table 15-10. Remote Enable Flags Values**

Bit Range	Function
0..3	Remote Enable Flags

Bit Range	Function	
	Value	Enumeration
	0x0	DISABLED
	0x7	TEMPORARILY LOCKED/DISABLED
	0xf	ENABLED REMOTE CONTROL
	0x1.	ENABLED REMOTE AND ENERGY CONTROL
	0x2..0x06, 0x8..0xe	Reserved
4..7	Device Status 2 Structure	
	Value	Enumeration
	0x0	PROPRIETARY
	0x1	PROPRIETARY
	0x2	IRIS SYMPTOM CODE
	0x3..0xf	Reserved

20288

**20289 ApplianceStatus2 Field**

20290 ApplianceStatus2 represents a detailed definition of Appliance state. If optionally provided, ApplianceStatus2 is  
20291 updated continuously as appliance state change.

20292 This field contains non-standardized or proprietary data. In the case of IRIS Symptom Code, 3 bytes representing the  
20293 3 digit encoding is provided (possibly complemented with proprietary bytes).

**20294 15.2.5.1.2 Effect on Receipt**

20295 On receipt of this command, the device is informed of a Household Appliance status.

**20296 15.2.5.2 Signal State Notification Command**

20297 This command SHALL be used to return household appliance status, automatically when appliance status changes.

**20298 15.2.5.2.1 Payload Format**

20299 The Signal State Notification Command payload SHALL be formatted as illustrated for the Signal State Response  
20300 Command Payload.

**20301 15.2.5.2.2 Effects on Receipt**

20302 On receipt of this command, the device is informed of a Household Appliance status.

**20303 15.2.6 Client**

20304 The client cluster has no dependencies or specific cluster attributes. The client side of this cluster receives the cluster  
20305 specific commands generated by the server. The client side of this cluster generates the cluster specific commands  
20306 received by the server as required by the application.



20307 **15.3 EN50523 Appliance Identification**

20308 **15.3.1 Overview**

20309 Please see section 2.2 for a general cluster overview defining cluster architecture, revision, classification,  
 20310 identification, etc.

20311 Attributes and commands for determining basic information about a device and setting user device information.

20312 The Appliance Identification Cluster is a transposition of EN50523 “Identify Product” functional block.

20313 **Note:** *Where a physical node supports multiple endpoints it will often be the case that many of these settings will*  
 20314 *apply to the whole node, that is they are the same for every endpoint on the device. In such cases they can be*  
 20315 *implemented once for the node, and mapped to each endpoint.*

20316 **15.3.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added; CCB 1893

20317 **15.3.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APLNCID	Type 2 (server to client)

20318 **15.3.1.3 Cluster Identifiers**

Identifier	Name
0x0b00	EN50523 Appliance Identification

20319 **15.3.2 Server**

20320 **15.3.2.1 Attributes**

20321 For convenience, the attributes defined in this specification are arranged into sets of related attributes; each set can  
 20322 contain up to 16 attributes. Attribute identifiers are encoded such that the most significant three nibbles specify the  
 20323 attribute set and the least significant nibble specifies the attribute within the set. The currently defined attribute sets  
 20324 are listed in Table 15-11.

20325 **Table 15-11. Appliance Identification Attribute Sets**

Attribute Set Identifier	Description
0x000	Basic Appliance Identification
0x001	Extended Appliance Identification

20326

20327 **15.3.2.2 Basic Appliance Identification Attribute Set**

20328 The Basic Appliance Identification attribute set contains the attributes summarized in Table 15-12.

20329 **Table 15-12. Attributes of the Appliance Identification Attribute Set**

Identifier	Name	Type	Range	Access	Def	M/O
0x0000	<i>BasicIdentification</i>	uint56	-	R	-	M

20330

20331 **15.3.2.3 BasicIdentification Attribute**20332 *BasicIdentification* is 56-bit bitmap (7 octets) and contains the basic appliance identification.20333 *BasicIdentification* is mandatory and must be included as part of the minimum data set to be provided by the household  
20334 appliance device.

20335 Table 15-13 provides attribute content specification.

20336 **Table 15-13. Basic Appliance Identification Content Specification**

Attribute Name	Field	Bits
<i>BasicIdentification</i>	Company ID	0x00-0x0f
	Brand ID	0x10-0x1f
	Product Type ID	0x20-0x2f
	Spec. Ver.	0x37-0x30

20337

20338 Table 15-13 provides Company ID and Brand ID fields content, according to [N2], Table 5.

20339 Table 15-14 provides Product Type IDs field content, again according to [N2] (see Table 6).

20340

**Table 15-14. Product Type IDs**

Device (Appliance)	Product Type ID
White Goods	0x0000
Dishwasher	0x5601
Tumble Dryer	0x5602

Device (Appliance)	Product Type ID
Washer Dryer	0x5603
Washing Machine	0x5604
Hobs	0x5E03
Induction Hobs	0x5E09
Oven	0x5E01
Electrical Oven	0x5E06
Refrigerator Freezer	0x6601

20341 **15.3.2.4 Extended Appliance Identification Attribute Set**

20342 The Extended Appliance Identification attribute set contains the attributes summarized in Table 15-15.

20343 **Table 15-15. Attributes of the Extended Appliance Identification Attribute Set**

Identifier	Name	Type	Range	Acc	Def	M/O
0x0010	<i>CompanyName</i>	string	0 to 16 Octets	R	-	O
0x0011	<i>CompanyId</i>	uint16	<i>all</i>	R	-	O
0x0012	<i>BrandName</i>	string	0 to 16 Octets	R	-	O
0x0013	<i>BrandId</i>	uint16	<i>all</i>	R	-	O
0x0014	<i>Model</i>	octstr	0 to 16 Octets	R	-	O
0x0015	<i>PartNumber</i>	octstr	0 to 16 Octets	R	-	O
0x0016	<i>ProductRevision</i>	octstr	0 to 6 Octets	R	-	O
0x0017	<i>SoftwareRevision</i>	octstr	0 to 6 Octets	R	-	O
0x0018	<i>ProductTypeName</i>	octstr	2 Octets	R	-	O
0x0019	<i>ProductTypeId</i>	uint16	<i>all</i>	R	-	O
0x001A	<i>CECEDSpecificationVersion</i>	uint8	<i>all</i>	R	-	O

20344

20345 **15.3.2.5 CompanyName Attribute**

20346 *CompanyName* is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates  
 20347 length) encoded in the UTF-8 format. Example Company Name labels are “Electrolux”, “Indesit Company”, “Candy”.  
 20348 The complete list of valid labels is defined in [E2], Table 7.

20349 **15.3.2.6 CompanyID Attribute**

20350 *CompanyID* is 16-bit in length unsigned integer which defines the appliance company identifier. The complete list of  
20351 valid company identifiers is defined in [E2], Table 7.

20352 **15.3.2.7 BrandName Attribute**

20353 *BrandName* is a ZCL Character String field capable of storing up to 16 character string (the first Octet indicates length)  
20354 encoded in the UTF-8 format. Example Brand Name labels are “Rex”, “Ariston”, “Hoover”. The complete list of valid  
20355 labels is defined in [E2], Table 7.

20356 **15.3.2.8 BrandID Attribute**

20357 *BrandID* is 16-bit in length unsigned integer which defines the appliance brand identifier. The complete list of valid  
20358 brand identifiers is defined in [E2], Table 7.

20359 Note that Brand Ids and Company Ids are independently defined. The advantage is that one brand of one producer  
20360 MAY have the same ID as a brand name of another producer.

20361 **15.3.2.9 Model Attribute**

20362 *Model* is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length) encoded  
20363 in the UTF-8 format. *Model* defines the appliance model name, decided by manufacturer.

20364 **15.3.2.10 PartNumber Attribute**

20365 *PartNumber* is a ZCL Octet String field capable of storing up to 16 character string (the first Octet indicates length)  
20366 encoded in the UTF-8 format. *PartNumber* defines the appliance part number, decided by manufacturer.

20367 **15.3.2.11 ProductRevision Attribute**

20368 *ProductRevision* is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length)  
20369 encoded in the UTF-8 format. *ProductRevision* defines the appliance revision code, decided by manufacturer.

20370 **15.3.2.12 SoftwareRevision Attribute**

20371 *SoftwareRevision* is a ZCL Octet String field capable of storing up to 6 character string (the first Octet indicates length)  
20372 encoded in the UTF-8 format. *SoftwareRevision* defines the appliance software revision code, decided by  
20373 manufacturer.

20374 **15.3.2.13 ProductTypeName Attribute**

20375 *ProductTypeName* is a 2 Octet in length String field which defines the appliance type label. Example  
20376 *ProductTypeName* labels are “WM”, “RE”, “GO”, respectively for Washing Machine, Refrigerator and Gas Oven.  
20377 The complete list of valid labels is defined in [E2], Table 8.

20378 **15.3.2.14 ProductTypeID Attribute**

20379 *ProductTypeID* is a 16-bit in length unsigned integer which defines the appliance type identifier. The structure and  
20380 complete list of valid *ProductTypeIDs* is defined in [E2], Table 7.

20381 **15.3.2.15 CECEDSpecificationVersion Attribute**

20382 *CECEDSpecificationVersion* is an 8-bit in length unsigned integer which defines the CECED reference  
 20383 documentation. Compliance and certification of appliance communication capabilities can be defined according to  
 20384 Table 15-16 (see [E2], Table 10).

20385 **Table 15-16. CECED Specification Version**

Specification Version	Value
Compliant with v1.0, not certified	0x10
Compliant with v1.0, certified	0x1A
Compliant with vX.0, not certified	0xX0
Compliant with vX.0, certified	0xXA

20386 **15.3.2.16 Commands Received**

20387 No cluster-specific commands are received by the server.

20388 **15.3.2.17 Commands Generated**

20389 No cluster-specific commands are generated by the server.

20390 **15.3.3 Client**

20391 The client cluster has no dependencies or cluster specific attributes. The client cluster has no cluster specific  
 20392 commands generated or received.

20393 **15.4 EN50523 Appliance Events and Alerts**

---

20394 **15.4.1 Overview**

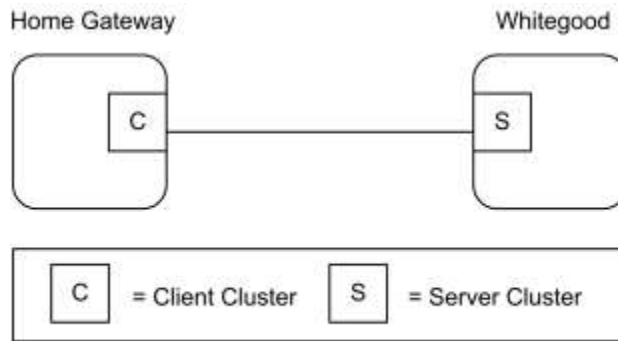
20395 Please see Chapter 2 for a general cluster overview defining cluster architecture, revision, classification, identification,  
 20396 etc.

20397 Attributes and commands for transmitting or notifying the occurrence of an event, such as “temperature reached” and  
 20398 of an alert such as alarm, fault or warning.

20399 It is based on the “Signal event” syntax of EN50523 and completed where necessary.

20400

**Figure 15-7. Typical Usage of the Appliance Events and Alerts Cluster**



*Note: Device names are examples for illustration purposes only*

20401

20402 There are two different types of occurrences: events and alerts.

20403 Each event is described through two fields:

- 20404 • An event header
- 20405 • An event identification value;

20406 The server notifies the client about the event occurred. There is no possibility for the client to get the event from the  
20407 server and to have a response.

20408 Each alert is described through three fields:

- 20409 • An alert identification value;
- 20410 • A category: either WARNING, DANGER, or FAILURE.
- 20411 • A presence/recovery flag, either the alert has been detected or the alert has been recovered.

20412 The server notifies the client regarding the alerts occurred. The client can also request the alerts from the server and  
20413 receive the related response.

### 20414 15.4.1.1 Revision History

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added

### 20415 15.4.1.2 Classification

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APPLEV	Type 2 (server to client)

### 20416 15.4.1.3 Cluster Identifiers

Identifier	Name
0x0b002	EN50523 Appliance Events and Alerts

20417 **15.4.2 Server**

20418 **15.4.2.1 Attributes**

20419 None

20420 **15.4.2.2 Commands Received**

20421 The received command IDs for the Appliance Events and Alerts Cluster are listed in Table 15-17.

20422 **Table 15-17. Received Commands IDs for the Events and Alerts Cluster**

Command Identifier Field Value	Description	M/O
0x00	Get Alerts	M

20423

20424 **15.4.2.2.1 Get Alerts Command**

20425 This basic message is used to retrieve Household Appliance current alerts.

20426 **15.4.2.2.1.1 Payload Format**

20427 This command does not have a payload.

20428 **15.4.2.2.1.2 Effects on Receipt**

20429 On receipt of this command, the device SHALL generate a Get Alerts Response command.

20430 **15.4.2.3 Commands Generated**

20431 The generated command IDs for the Appliance Events and Alerts Cluster are listed in Table 15-18.

20432 **Table 15-18. Generated Commands IDs for the Appliance Events and Alerts Cluster**

Command Identifier Field Value	Description	M/O
0x00	Get Alerts Response	M
0x01	Alerts Notification	M
0x02	Event Notification	M

20433 **15.4.2.3.1 Get Alerts Response Command**

20434 This message is used to return household appliance current alerts.

20435 **15.4.2.3.1.1 Payload Format**

20436 The payload SHALL be formatted as illustrated in Figure 15-8.

20437

**Figure 15-8. Format of the Get Alerts Response Command Payload**

<b>Octets</b>	1	3	...	3
<b>Data Type</b>	uint8	uint24	...	uint24
<b>Field Name</b>	Alerts Count <sup>164</sup>	Alert structure 1	...	Alert structure <i>n</i>

20438 **15.4.2.3.1.1.1 Payload Details**

20439 The **Alerts Count** field: the data field is an 8 bits in length unsigned integer, containing the following alerts structures  
20440 count and alert structure type.

20441 Table 15-19 provides details about Alerts Count and Structure field organization.

20442 **Table 15-19. Alert Count Organization**

<b>Bit range</b>	<b>Function</b>	
0..3	Number of Alerts <i>n</i>	
4..7	Type of alert	
	<b>Value</b>	<b>Enumeration</b>
	0x0 0x1..0xf	UNSTRUCTURED Reserved

20443

<sup>164</sup> Even if the ApplianceAlertList array number of element field is 16-bit in length, the actual content is limited to 0x000*n*, where, in actual implementations, *n* is lower than 255 (except for the invalid condition, 0xffff). Then, the notification of the Alert count is mapped to a single byte (following appliance interworking specifications).



20444 Each *Alerts Structure* field SHALL be formatted as illustrated in Table 15-20.

20445 **Table 15-20. Alerts Structure Organization**

Bit range	Function	
0..7	Alert id	
8..11	Category	
	Value	Enumeration
	0x0 0x1 0x2 0x3 0x4 – 0xf	Reserved WARNING DANGER FAILURE Reserved
12..13	Presence recovery	
	Value	Enumeration
	0x0 0x1 0x2 – 0x3	RECOVERY PRESENCE Reserved
16..23	Manufacturer specific bits	

20446

20447 The *Alert ID* field can have the following values:

- 20448
- Value 0 is reserved.
- 20449
- Values ranging from 1 to 63 are standardized.
- 20450
- Values ranging from 64 to 127 are reserved.
- 20451
- Values ranging from 128 to 255 are manufacturer specific.

20452 **15.4.2.3.1.2 Effects on Receipt**

20453 On receipt of this command, the device is informed of a Household Appliance warning and fault occurrence.

20454 **15.4.2.3.2 Alerts Notification Command**

20455 This message is used to notify the current modification of warning and/or fault conditions.

20456 **15.4.2.3.2.1 Payload Format**

20457 The payload SHALL be formatted as illustrated in Figure 15-9.

20458

**Figure 15-9. Format of the Alerts Notification Command Payload**

<b>Octets</b>	1	3	...	3
<b>Data Type</b>	uint8	uint24	...	uint24
<b>Field Name</b>	Alerts Count	Alert structure 1	...	Alert structure <i>n</i>

20459 **15.4.2.3.2.1 Payload Details**

20460 See Get Alert Response command.

20461 **15.4.2.3.2.2 Effects on Receipt**

20462 On receipt of this command, the device is informed of a Household Appliance warning and fault occurrence.

20463 **15.4.2.3.3 Event Notification Command**

20464 This message is used to notify an event occurred during the normal working of the appliance.

20465 **15.4.2.3.3.1 Payload Format**

20466 The payload SHALL be formatted as illustrated in Figure 15-10.

20467

**Figure 15-10. Format of the Event Notification Command Payload**

<b>Octets</b>	1	1
<b>Data Type</b>	uint8	uint8
<b>Field Name</b>	Event Header	Event Identification

20468 **15.4.2.3.3.1.1 Payload Details**20469 The *Event Header* is a reserved field set to 0.20470 The *Event Identification* field: the *Event Identification* is an 8-bits in length field identifying the event to be notified.

20471 The codes used for this field SHALL match those shown in Table 15-21:

20472

**Table 15-21. Event Identification**

<b>Event Identification</b>	<b>Value</b>	<b>Description</b>
END_OF_CYCLE	0x01	End of the working cycle reached
TEMPERATURE_REACHED	0x04	Set Temperature Reached
END_OF_COOKING	0x05	End of cooking process
SWITCHING OFF	0x06	
<i>Manufacturer Specific</i>	0x40-0xf6	Manufacturer specific Id range
WRONG_DATA	0xf7	

Event Identification	Value	Description
<i>Manufacturer Specific</i>	0xf8-0xff	Manufacturer specific Id range

20473 **15.4.2.3.3.2 Effects on Receipt**

20474 On receipt of this command, the device is informed of a Household Appliance working event occurrence.

20475 **15.4.3 Client**

20476 The client cluster has no dependencies or specific cluster attributes. The client side of this cluster receives the cluster  
 20477 specific commands generated by the server. The client side of this cluster generates the cluster specific commands  
 20478 received by the server as required by the application.

20479 **15.5 Appliance Statistics**

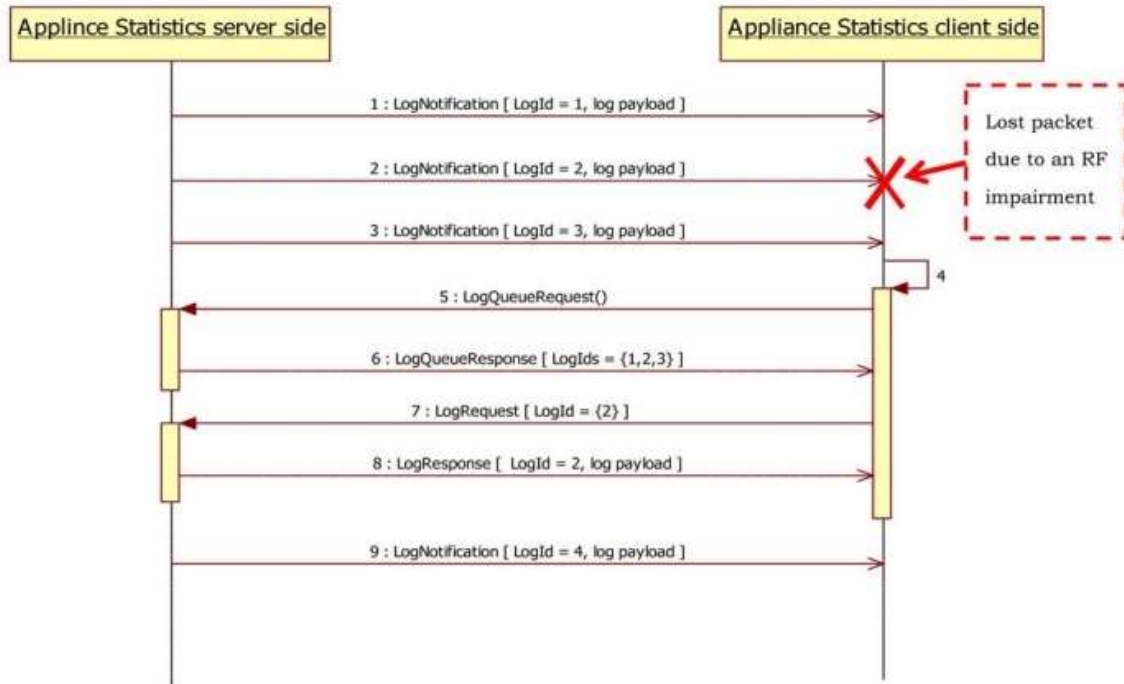
---

20480 **15.5.1 Overview**

20481 This cluster provides a mechanism for the transmitting appliance statistics to a collection unit (gateway). The statistics  
 20482 can be in format of data logs. In case of statistic information that will not fit the single payload, the Partition cluster  
 20483 SHOULD be used.

20484 Each appliance uses persistent memory to temporarily store collected statistic logs (entries). The  
 20485 maximum number of stored statistic logs is appliance dependent. If some log notification packets are lost  
 20486 due to temporary unreliable RF communication, sequential Log IDs allow the collection of the missing  
 20487 logs. The following is a simple example of an application-level policy used for log collection. When  
 20488 receiving logs with non-consecutive Log IDs, the client can ask server side for the available log queue to  
 20489 verify the actual availability of the missing log. If present, the log can be explicitly retrieved using the  
 20490 LogRequest command.

20491



20492

20493 **15.5.1.1 Revision History**

Rev	Description
1	mandatory global <i>ClusterRevision</i> attribute added;CCB 1893

20494 **15.5.1.2 Classification**

Hierarchy	Role	PICS Code	Primary Transaction
Base	Application	APPLST	Type 2 (server to client)

20495 **15.5.1.3 Cluster Identifiers**

Identifier	Name
0x0b003	EN50523 Appliance Statistics

20496 **15.5.2 Server**

20497 **15.5.2.1 Attributes**

20498 The server side of this cluster contains the attributes the statistics and log information shown in Table 15-22.

20499

**Table 15-22. Server Attributes**

Identifier	Description	Type	Access	Default	M/O
0x0000	<i>LogMaxSize</i>	uint32	R	0x0000003C	M
0x0001	<i>LogQueueMaxSize</i>	uint8	R	0x01	M

20500 **15.5.2.1.1 *LogMaxSize* Attribute**

20501 The *LogMaxSize* attribute describes the maximum size of a log payload that can be transferred using the Log  
 20502 Notification and Log Response commands. In case the *LogMaxSize* attribute is greater than 70 bytes (0x46) the  
 20503 Appliance Statistics commands SHOULD be transferred using the partition cluster. This is the case of a “bulk log”  
 20504 transferred from a server side (e.g., White Goods) to a client side (e.g., home gateway) of the Appliance Statistics  
 20505 Cluster.

20506 **15.5.2.1.2 *LogQueueMaxSize* Attribute**

20507 The *LogQueueMaxSize* attribute describes the maximum number of logs that are available in the server side of the  
 20508 Appliance Statistics cluster. The logs MAY be retrieved by the client using the Log Request command.

20509 **15.5.2.2 Commands**

20510 The generated command IDs for the Appliance Statistics Server are listed in Table 15-23.

20511 **Table 15-23. Commands Generated by the Appliance Statistics Server**

Command ID	Description	M/O
0x00	Log Notification	M
0x01	Log Response	M
0x02	Log Queue Response	M
0x03	Statistics Available	M

20512 **15.5.2.2.1 Log Notification**

20513 The Appliance Statistics Cluster server occasionally sends out a Log Notification command to the devices to which it  
 20514 needs to log information related to statistics (e.g., home gateways) which implement the client side of Appliance  
 20515 Statistics Cluster.

20516 **15.5.2.2.1.1 Payload Format**

20517

**Figure 15-11. Format of the Log Notification Payload**

<b>Octets</b>	4	4	4	1	...	1
<b>Data Type</b>	UTC	uint32	uint32	data8	data8	data8
<b>Field Name</b>	Time Stamp	Log ID	Log Length	Log Payload		

20518 **15.5.2.2.1.2 When Generated**

20519 The Log Notification command is generated when the appliance needs to send log information related to its statistics  
20520 to a remote device (e.g., home gateway) without being solicited by the client side. The log information sent with the  
20521 Log Notification command from the server side is not solicited by specific command generated by the client side of  
20522 the Appliance Statistics cluster. The Log ID field identifies uniquely the log information contained in the log payload.  
20523 Log IDs SHALL be consecutive. Log Length field indicated the length in bytes of the log payload and SHALL be less  
20524 than *LogMaxSize* attribute.

20525 If the device generating the Log Notification command is not able to generate the time stamp information it SHALL  
20526 insert an invalid UTC Time (0xffffffff). In this case the server side of the Appliance statistics cluster (e.g., a home  
20527 gateway) SHOULD insert a timestamp of the received log notification if available before storing or transmitting the  
20528 log information to backend systems.

20529 **15.5.2.2.1.3 Effect Upon Receipt**

20530 Upon receipt of the Log Notification command, the Appliance statistics client will respond with a Default Response  
20531 command if requested or if an error occurs. In case of error the server side of Appliance statistics cluster MAY store  
20532 the information in the queue and notify the client that there are statistics available by using the Statistic Available  
20533 command.

20534 **15.5.2.2.2 Log Response**

20535 The Appliance Statistics Cluster server sends out a Log Response command to respond to a Log Request command  
20536 generated by the client side of the Appliance Statistics cluster.

20537 **15.5.2.2.2.1 Payload Format**

20538 The payload of the Log Response command is the same as the Log Notification command.

20539 **15.5.2.2.2.2 When Generated**

20540 The Log Response command is generated to respond to Log Request sent from a device supporting the client side of  
20541 the Appliance Statistics cluster (e.g., home gateway).

20542 **15.5.2.2.2.3 Effect Upon Receipt**

20543 Upon receipt of the Log Response command, the Appliance statistics client will respond with a Default Response  
20544 command if requested or if an error occurs.

20545 **15.5.2.2.3 Log Queue Response**

20546 The Log Queue Response command is generated as a response to a Log Queue Request command in order to notify  
20547 the client side of the Appliance statistics cluster about the logs stored in the server side (queue) that can be retrieved  
20548 by the client side of this cluster through a Log Request command. Please note that the *LogQueueSize* field SHALL  
20549 be less than the *LogQueueMaxSize* attribute.

20550 **15.5.2.2.3.1 Payload Format**

20551

**Figure 15-12. Format of the Log Queue Response Payload**

<b>Octets</b>	1	4	4	4
<b>Data Type</b>	uint8	uint32	...	uint32
<b>Field Name</b>	Log Queue Size	Log ID	...	Log ID

20552 **15.5.2.2.3.2 When Generated**

20553 The Log Queue Response command is generated in response to a Log Queue Request sent from a device supporting  
 20554 the client side of the Appliance Statistics cluster (e.g., home gateway) Please note that if Log Queue Size is equal to  
 20555 zero (not logs in the queue), the packet SHALL not carry Log IDs.

20556 **15.5.2.2.3.3 Effect Upon Receipt**

20557 Upon receipt of the Log Queue Response command, the Appliance statistics client will respond with a Default  
 20558 Response command if requested or if an error occurs. The client side of the appliance statistics willing to get the logs  
 20559 in the queue SHALL then use only the Log IDs that have been indicated in the Log Queue Response.

20560 **15.5.2.2.4 Statistics Available**

20561 The Appliance Statistics Cluster server sends out a Statistic Available command to notify the client side of the  
 20562 Appliance Statistics cluster that there are statistics that can be retrieved by using the Log Request command.

20563 **15.5.2.2.4.1 Payload Format**

20564 The Statistic Available command is the same as the Log Queue Response command. The Log IDs that can be retrieved  
 20565 by the client are indicated in the payload.

20566 **15.5.2.2.4.2 When Generated**

20567 The Statistic Available command is generated to notify a device supporting the client side of the Appliance Statistics  
 20568 cluster (e.g., home gateway) to get the statistics information from the log queue as soon as available to perform this  
 20569 operation.

20570 **15.5.2.2.4.3 Effect Upon Receipt**

20571 Upon receipt of the Statistic Available command, the client side of the Appliance Statistics cluster is notified on the  
 20572 availability of statistics in the server side that can be retrieved by using Log Request commands.

20573 The Appliance statistics client will respond with a Default Response command if requested or if an error occurs.

20574 **15.5.3 Client**

20575 **15.5.3.1 Attributes**

20576 There are no attributes on the client side of the Appliance Statistics Cluster.

20577 **15.5.3.2 Commands**

20578 The generated command IDs for the Appliance Statistics Client are listed in Table 15-24.

20579

**Table 15-24. Commands Generated by the Appliance Statistics Client**

Command ID	Description	M/O
0x00	Log Request	M
0x01	Log Queue Request	M

20580

### 20581 **15.5.3.2.1 Log Request**

20582 The Log Request command is send from a device supporting the client side of the Appliance Statistics cluster (e.g.,  
20583 Home Gateway) to retrieve the log from the device supporting the server side (e.g., appliance).

#### 20584 **15.5.3.2.1.1 Payload Format**

20585

**Figure 15-13. Format of the Log Request Payload**

<b>Octets</b>	4
<b>Data Type</b>	uint32
<b>Field Name</b>	Log ID

#### 20586 **15.5.3.2.1.2 When Generated**

20587 The Log Request command is generated to retrieve a log information from a device supporting the server side of the  
20588 Appliance Statistics cluster (e.g., appliance). The log information is addressed by referencing it with the Log ID field.  
20589 In order to get the Log ID that can be retrieved with the Log Request command, the Log Queue Request command  
20590 MAY be used.

#### 20591 **15.5.3.2.1.3 Effect Upon Receipt**

20592 Upon receipt of the Log Request command, the Appliance statistics server will respond with a Log Response command  
20593 if the log is available or with a Default Response if an error occurs. In case the Log ID is not available in the server  
20594 side of the cluster the status code carried by the Default Response SHALL be “NOT\_FOUND.”

### 20595 **15.5.3.2.2 Log Queue Request**

20596 The Log Queue Request command is sent from a device supporting the client side of the Appliance Statistics cluster  
20597 (e.g., Home Gateway) to retrieve the information about the logs inserted in the queue, from the device supporting the  
20598 server side (e.g., appliance).

#### 20599 **15.5.3.2.2.1 Payload Format**

20600 The Log Queue Request command has no payload.

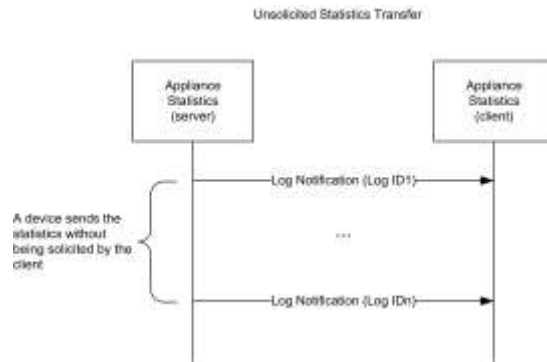
## 20601 **15.5.4 Appliance Statistics Cluster Sequence Diagram**

20602 Figure 15-14 shows a typical sequence interaction between the client and server sides of the Appliance Statistics  
20603 Cluster.

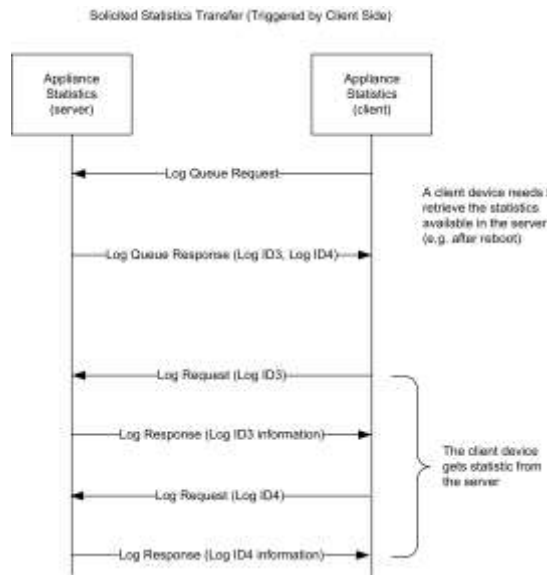


20604

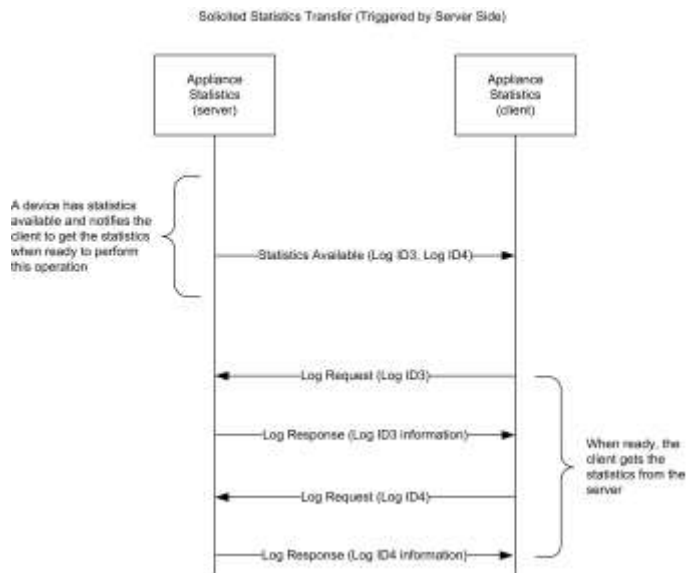
**Figure 15-14. Appliance Statistics Cluster Sequence Diagram**



20605



20606



20607

20608

20609