



1

# **ZCLIP Base Device Behavior Specification Version 1.0**

Zigbee Document 16-07008-075

Feb. 04, 2019

Sponsored by: Zigbee Alliance

Accepted by                      This document has been accepted for release by the ZigBee Alliance Board of Directors

Abstract                         This specification defines the ZCL over IP base device behavior specification for devices operating on an IP based stack.

Keywords                        Base device, ZCLIP

2

---

Copyright © Zigbee Alliance, Inc. (1996-2019). All rights reserved.

508 Second Street, Suite 206 Davis, CA 95616 - USA

<http://www.Zigbee.org>

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.

3

This page is intentionally blank

## Notice of Use and Disclosure

Copyright © Zigbee Alliance, Inc. (1996-2019). All rights Reserved. This information within this document is the property of the Zigbee Alliance and its use and disclosure are restricted.

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

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

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

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



## Revision History

Revision	Date	Details	Editor
000	March 1 <sup>st</sup> , 2016	Initial draft for comment	Robert Cragie
001	May 6 <sup>th</sup> , 2016	Added Comment text for comments: 90, 96, 104, 105, 111, 122, 85, 134	Ezra Hale
008	July 25 <sup>th</sup> , 2016	Created from 16-07008-007.	Phil Jamieson
009	July 28, 2016	Added Security section	Skip Ashton
010	August 8, 2016	Added Groups, Reporting and Discovery sections	Ezra Hale
011	August 8, 2016	Updated versioning to be correct	Ezra Hale
012	August 8, 2016	Updated to include ABNF and fix numbering	Ezra Hale
013	August 9, 2016	Added UID comments to binding, EZ-Mode and COAP response code sections	Ezra Hale
014	August 10, 2016	Added comment to filters	Ezra Hale
015	August 16, 2016	Added comments from review	Ezra Hale
016	August 16, 2016	Further updates to comments from review	Ezra Hale
017	August 17, 2016	Updated following meeting with the group: removed ABNF grammar, accepted changes, updated tables, removed examples to appendix, removed non-spec sections, removed accepted and reviewed comments.	Ezra Hale
018	August 19, 2016	Added section on URI format from Jason	Ezra Hale
019	August 22, 2016	Fixed typo in groups section	Ezra Hale
020	August 22, 2016	Updated with comments from Luca	Ezra Hale
021	August 22, 2016	PDF for 0.7 Balloting	Ezra Hale
022	October 5, 2016	Editorial revisions per 0.7 Ballot comments:  447, 448, 451, 455, 456, 457, 461, 463, 465, 466, 529, 530, 531, 534, 557, 559, 592, 603, 616, 636, 637, 638, 639, 641, 642, 653, 655, 656, 657, 658, 660, 661, 662, 663, 664, 668, 672, 673, 674, 722, 725, 730, 741, 746, 761, 787	Jason Perreault
023	December 1, 2016	Substantial document reorganization guided by David Smith's Comment 527 proposal.  FRONT MATTER Added Scope and Purpose based on NFR. Added some content for Acronyms and Definitions. Reorganized References and acronym-based shorthand (e.g. [COAP], not [RFC7252]). Added a few more references. Added Notational and Descriptive Conventions.  FOUNDATION Added Overview with ZCL Client/Server text and figure taken from [ZCL]. Reorganized Foundation section with placeholder headings and/or inline notes about possible topic/content to be added. Updates to Groups and Reporting sections from	Jason Perreault

	<p>submitted comments/text. Added UID section. ZCLIP Messaging contains descriptions for URI Format, Payload, and Common Message Processing Rules. Handling of requests received via multicast phrased in terms of differences to rules for unicast.</p> <p><b>ZCLIP RESOURCES</b> Segregates URI Request/Response descriptions from rest of normative text. Resources classified as List or Instance (occasionally Resource). Descriptions condensed to omit redundancies with common message processing rules. Simple Descriptor is described. Attribute query parameter description revised. Attribute value error checking in its own section and referenced where needed. Binding, Report Configuration, and Notification resource descriptions significantly revised per workgroup discussion and draft text submission. Group Cluster-related sections moved to Annex. A lot of empty subsections or unpolished text remains for Command, Group, and EZ Mode resources.</p> <p><b>DISCOVERY, EZ MODE, SECURITY</b> The many examples in Discovery are moved to an Annex section. Otherwise no significant changes or additions.</p> <p><b>ZCLIP Cluster Translation Rules</b> New placeholder section with a few inline notes to describe how to translate ZCL cluster definition into ZCLIP form.</p> <p><b>ANNEXES</b> Not numbered differently yet (Word issue). Discovery examples moved here. CoAP Response Code table moved here and restructured. Reference Device for Test added. ZCL General Request Command Mapping added. Cluster Implementations added, Group Cluster description moved here.</p>	
024      December 15, 2016	<p><b>FOUNDATION</b> CoAP Accept Option is nonrepeatable, only one format may be specified.</p> <p><b>RESOURCES</b> Description of Command resources. Description of Group resources.</p> <p><b>DISCOVERY</b> Substantially rewritten for consistency of language with rest of document and to recast description similar to common message processing rules. Consolidate separate sections concerning single- and multi-attribute query filter.</p> <p><b>ZCL CLUSTER TRANSLATION</b> initial text with general rules for translating a ZCL cluster definition into ZCLIP form.</p> <p><b>MISCELLANEOUS</b> Consistent header style for TOC/Figures/Tables. Table of Contents has more content. Annex section numbering corrected. Empty annex for ZCL/ZCLIP differences added. Annexes reordered.</p>	Jason Perreault

025	January 31, 2017	<p>Addressed Comments 470, 471, 507, 573 for common understanding for next test event.</p> <p>470: Change payload structure for Attribute retrieval to allow reporting of ZCL Status per attribute instead of attribute value, if value is inaccessible. Common payload format for both filtered multi-attribute and single attribute instance resources. Updates to both endpoint and group attribute resource descriptions.</p> <p>471: Specify sizes for Binding ID and Report ID at 8 bits with range 0x00-0xFE; 0xFF is reserved.</p> <p>507: Clarification of Default Response. See adds/edits under Transactions, ZCLIP payload data representation, and common message processing rules.</p> <p>573: Prohibit IPv6 address as URI-Host in unicast bindings. Must be static host identifier (hostname or UID). This changes URI-Host from "MAY omit" to "SHALL contain" since the unicast source IPv6 address of the POST-er is not applicable as a default.</p> <p>Remove Binding Instance PUT error checks that prohibited changing an existing binding from unicast to multicast or vice versa.</p>	Jason Perreault
026	February 1, 2017	<p>Reconsideration of 573: Instead of prohibiting unicast IPv6 address in binding URI-Host, change from SHALL NOT to SHOULD NOT in 3.10.3.1. Delete corresponding SHALL NOT error checking in Binding Collection POST and Binding Instance PUT response descriptions in 3.10.4 and 3.11.4.</p>	Jason Perreault
027	May 1, 2017	<p>More than 60 comments addressed with varying degrees of success.</p>	Jason Perreault
028	May 15, 2017	<p>Accept all tracked changes from 027.</p> <p>Replace Security chapter with submission for lean token per 17-07018-005.</p> <p>Replace EZ-Mode chapter with prior submission for comment 525 per 16-07038-000.</p> <p>Update description of mapping Group ID to IPv6 multicast address per 2017-05-03 entries for comment 538 per 16-07054-001.</p>	Jason Perreault
029	May 15, 2017	Initial PDF.	Jason Perreault
030	May 15, 2017	Corrected PDF (doc version)	Jason Perreault
031	July 3, 2017	<p>Interim revision to address many minor and primarily editorial comments from r030 re-ballot: typos, blank rows in tables, incorrect sections references, etc. All prior change tracking is accepted / cleared. More substantive comments will be addressed in subsequent revisions.</p>	Jason Perreault
032	July 6, 2017	<p>Another interim revision to address more editorial and some simple substantive comments. Accept tracked changes and clear inline comments for 031 edits accepted by the submitter, retain other tracked edits and inline comments from r031.</p>	Jason Perreault
033	September 5, 2017	<p>Updated per 0.7 re-ballot comment resolutions in preparation for September 2017 test event.</p>	Jason Perreault

		<p>In addition to the following highlights, see tracked changes / margin comments for other minor edits.</p> <p>"ZCLIP" used consistently in place of most "ZCL over IP" et al.</p> <p>CoAP token, codes, options, block transfer descriptions are improved.</p> <p>Description of IPv6 Multicast usage is improved.</p> <p>UID to IPv6 address resolution is described.</p> <p>Revised constraints on CBOR map keys.</p> <p>Require CoAP Content-Format option to be included in messages.</p> <p>Corrected description of how a UID is represented in the Uri-Host portion of a URI (binding, notification).</p> <p>Notification message now includes Binding ID.</p> <p>Discovery sections and examples changed to reflect application/link-format+cbor Content-Format in message payload.</p> <p>Improved treatment of discovery link attributes.</p> <p>Security chapter updated for unicast v. multicast considerations.</p>	
034	September 6, 2017	<p>Incorporate edits from 17-07026-001-Edits_to_multicast_groups.docx to address Comments 1136, 1256, 1263.</p> <p>Incorporate edits from 17-07008-004-ZCLIP_metadata_section_0317.docx to address Comments 1187, 1249, 1257.</p> <p>Revised Resource Directory message description for link-format+cbor. Added note in Discovery chapter that Content-Format value 65064 would be used as link-format+cbor for development pending IANA's assignment of actual value.</p> <p>Distinguished 5.00 and 5.03 CoAP response codes in message processing rules and Annex table.</p>	Jason Perreault
035	October 16, 2017	Incorporation of edits assigned to Jason against R16. Resolution of edits from Eindhoven test event.	Ezra Hale
036	October 18, 2017	Ongoing edits on access examples	Ezra Hale
037	October 25, 2017	Merge in Security section to cover GET and DELETE operations.	Ezra Hale
038	November 15, 2017	Comments addressed: 01213, 01345, 01105, 01279,	Ezra Hale
039	November 15, 2017	Comments addressed: 1214	Ezra Hale
040	November 15, 2017	Frozen and PDF created for gating test event #2 in Nice, France, Dec 4-8, 2017	Ezra Hale
041	December 5, 2017	<p>Properly merged edits from comment 1074.</p> <p>Edits from gating test event #2 in Nice, France, Dec. 4-8, 2017</p>	Ezra Hale
042	December 20, 2017	Edits per group resolution of Comments: 1327 RD Discovery Mismatch	Jason Perreault



		1330 Multiple Resource Directories 1333 Sleepy Devices / Proxies Regen TOC, List of Figures, List of Tables.	
045	January 18, 2018	Updated with security changes from Sander to resolve 1338	Ezra Hale
046	January 18, 2018	PDF of spec frozen for Gating test event #3 Boston, MA	Ezra Hale
047	January 18, 2018	Changes from Gating test event #3, Boston, MA	Ezra Hale
048	February 1, 2018	Changes from Gating test event #3, Boston, MA	Ezra Hale
049	February 28, 2018	Update for comments 1460, 1464	Ezra Hale
050	March 06, 2018	Update for comment 1462	Ezra Hale
051	March 09, 2018	Fixed link error in security section by replacing with content from document 17-07044 Comments: 1423, 1424, 1425, 1427	Ezra Hale
052	March 09, 2018	Comments: 1507, 1504, 1502, 1500, 1498, 1496, 1493, 1492, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485	Ezra Hale
053	March 14, 2018	Comments: 1488, 1510, 1511, 1513, 1514, 1516, 1515, 1512, 1509, 1508, 1506, 1505, 1503, 1497, 1495, 1594, 1490, 1486, 1487, 1489, 1545, 1561, 1563, 1560, 1564, 1547, 1555, 1565, 1562, 1557, 1556	Ezra Hale
054	March 15, 2018	Comments: 1552, 1548, 1537, 1536, 1538, 1532, 1526, 1525, 1521, 1413, 1414, 1417, 1421, 1416, 1426, 1520, 1522, 1527,	Ezra Hale
055	March 20, 2018	Fix broken links in Security Section	Ezra Hale
056	March 21, 2018	Comments: 1566, 1540, 1523, 1528, 1529	Ezra Hale
057	March 23, 2018	Comments: 1419, 1530, 1534, 1415, 1539, 1535, 1533, 1184, 614, 1219, 1033	Ezra Hale
058	March 23, 2018	Comments: 1457, Minor updates following PICS review.	Ezra Hale
059	March 27, 2018	Merged document with RD security changes from Luca, Comment 1239	Ezra Hale
060	March 28, 2018	Comments: 1519, 1570, 1524, 1458, Updated Figure 3 with image from Luca. Spec version to be used in Gating Test Event #4, Eindhoven April 10 – 13.	Ezra Hale
061	April 5, 2018	Comments: 1518, 1572, 1554, Added Group examples provided by David Smith. Reviewed with group April 5, 2018	Ezra Hale
062	April 10, 2018	Comments from Gating Test Event #4, Eindhoven April 10, 2018	Ezra Hale
063	April 11, 2018	PDF of doc for PICS development	Ezra Hale
064	May 17, 2018	Ongoing comment resolution from test event in Eindhoven, Comments resolved in preparation for 0.9 letter ballot.	Ezra Hale
065	May 22, 2018	PDF version of R64 for 0.9 Letter Ballot	Ezra Hale

066	June 20, 2018	Word Doc for post 0.9 Letter Ballot, edits coming out of 0.9 letter ballot.	Ezra Hale
067	August 20, 2018	Edits from 0.9 letter ballot and Gating Test Event #5 in Toronto 07/20/2018	Ezra Hale
068	August 29, 2018	Resolve comments in preparation for 0.9 Submission to the TC	Ezra Hale
069	August 30, 2018	PDF for submission of 0.9 to the TC	Ezra Hale
070	October 26, 2018	Updates from GTE #6, comment resolution done at the Athens AMM	Ezra Hale
071	November 01, 2018	Updates from comments prior to VTE taken out of meeting minutes.	Ezra Hale
072	November 27, 2018	Updated following meeting with David Smith on VTE #7	Ezra Hale
073	December 11, 2018	PDF R72 for 1.0 application to the ERP and TC	Ezra Hale
074	February 04, 2019	Final Word Document for 1.0	Ezra Hale
075	February 04, 2019	Final PDF for 1.0	Ezra Hale

32

## Participants

Shane Almeida	Alin Lazar
Victor Berrios	Sergei Lissiano
Gary Butt	Sebastien Marion
Lee Byrd	Jason Perreault
Gatien Chapon	Sander Raaijmakers
Michael Cowan	Brad Ree
Robert Cragie	Luc Revardel
Bruno DeSmet	David Smith
Andras Fekete	Igor Vartanov
Ezra Hale	Jordan Wills
Phil Jamieson	Luca Zappaterra
Lars Karlstrom	Teresa Zotti

33

# Table of Contents

34	<b>Table of Contents</b>		
35	Notice of Use and Disclosure .....		3
36	Revision History .....		5
37	Participants .....		11
38	Table of Contents.....		12
39	List of Figures.....		21
40	List of Tables.....		23
41	1 Introduction .....		26
42	1.1 Scope .....		26
43	1.2 Purpose .....		26
44	1.3 Conformance testing.....		26
45	1.4 Acronyms and Abbreviations .....		27
46	1.5 Definitions .....		28
47	1.6 References .....		29
48	1.6.1 Zigbee Alliance Documents .....		29
49	1.6.2 IETF Documents .....		29
50	1.6.3 IEEE Documents .....		31
51	1.6.4 Thread Group Documents .....		31
52	1.6.5 Fairhair Alliance Documents .....		31
53	1.6.6 NIST Documents .....		31
54	1.7 Notational and Descriptive Conventions .....		31
55	1.7.1 Identifiers .....		31
56	1.7.2 Descriptive References to Identifier Instances .....		32
57	1.7.3 Message Payload Contents.....		32
58	1.7.4 Key Words .....		32
59	2 ZCLIP Foundation.....		33
60	2.1 ZCL Overview.....		33
61	2.1.1 System Architecture .....		33
62	2.1.2 ZCL Client/Server Model .....		34
63	2.1.3 Endpoints .....		34
64	2.1.4 Clusters .....		34
65	2.1.5 Attributes.....		35
66	2.1.6 Commands .....		35
67	2.1.7 Groups.....		35
68	2.1.8 Bindings .....		35

69	2.1.9	Reports.....	36
70	2.1.10	Manufacturer Extensions .....	36
71	2.1.11	Persistent Data .....	36
72	2.1.12	Minimal Requirements for All Devices .....	36
73	2.2	Protocol Layers .....	37
74	2.2.1	IP.....	37
75	2.2.2	UDP .....	37
76	2.2.3	CoAP .....	37
77	2.3	Transactions .....	39
78	2.3.1	CoAP Message Types for ZCL General Commands .....	39
79	2.3.2	ZCL Default Response Command .....	41
80	2.4	Bindings .....	42
81	2.4.1	Binding ID .....	42
82	2.5	Groups.....	42
83	2.5.1	Group ID.....	43
84	2.5.2	Broadcast Group .....	43
85	2.5.3	Support for Groups Cluster.....	43
86	2.5.4	No Default Response to Group Multicast .....	43
87	2.5.5	Group ID Mapping to Multicast IPv6 Address .....	43
88	2.5.6	Assignment of IPv6 Multicast Addresses .....	45
89	2.6	Reporting.....	45
90	2.6.1	Report ID .....	45
91	2.6.2	Default Report .....	45
92	2.6.3	Notification Generation .....	46
93	2.6.4	Group Support .....	46
94	2.7	Unique Identifier (UID) .....	47
95	2.7.1	UID Value Generation.....	47
96	2.7.2	UID Resolution.....	48
97	2.8	ZCLIP Messaging .....	49
98	2.8.1	URI Format.....	49
99	2.8.2	Payload Data Representation .....	56
100	2.8.3	Payload Data Size .....	60
101	2.8.4	ZCL Status Codes .....	60
102	2.8.5	Backward Compatible Request and Response Handling .....	68
103	2.8.6	Common Message Processing Rules .....	69
104	2.8.7	TLV Extensions .....	73
105	3	ZCLIP Resources .....	74
106	3.1	ZCL Entry Point Resources.....	74

107	3.1.1	GET Request .....	74
108	3.1.2	GET Response .....	74
109	3.1.3	ZCL Entry Point Resources Access Example .....	74
110	3.2	Endpoint Collection .....	75
111	3.2.1	GET Request .....	75
112	3.2.2	GET Response .....	75
113	3.2.3	Endpoint Collection Access Example .....	75
114	3.3	Endpoint Resource Collection .....	75
115	3.3.1	GET Request .....	75
116	3.3.2	GET Response .....	76
117	3.3.3	Endpoint Resource Collection Access Example .....	76
118	3.4	Cluster Resource Collection .....	76
119	3.4.1	GET Request .....	76
120	3.4.2	GET Response .....	76
121	3.4.3	Cluster Resource Collection Access Example .....	77
122	3.5	Attribute Collection .....	77
123	3.5.1	Query Parameters .....	77
124	3.5.2	Attribute Value Error Checking .....	78
125	3.5.3	GET Request .....	79
126	3.5.4	GET Response .....	79
127	3.5.5	POST Request .....	80
128	3.5.6	POST Response .....	80
129	3.5.7	Attribute Collection Access Example .....	80
130	3.6	Attribute Instance .....	81
131	3.6.1	GET Request .....	81
132	3.6.2	GET Response .....	81
133	3.6.3	PUT Request .....	81
134	3.6.4	PUT Response .....	81
135	3.6.5	Attribute Instance Access Example .....	82
136	3.7	Command Collection .....	82
137	3.7.1	GET Request .....	82
138	3.7.2	GET Response .....	83
139	3.7.3	Command Collection Access Example .....	83
140	3.8	Command Instance .....	83
141	3.8.1	Command/Response Payload Structure .....	83
142	3.8.2	POST Request .....	83
143	3.8.3	POST Response .....	84
144	3.8.4	Command Instance Access Example .....	84
145	3.9	Binding Collection .....	84

146	3.9.1	GET Request.....	85
147	3.9.2	GET Response .....	85
148	3.9.3	POST Request.....	85
149	3.9.4	POST Response .....	86
150	3.9.5	Binding Collection Access Example .....	86
151	3.10	Binding Instance.....	87
152	3.10.1	GET Request.....	87
153	3.10.2	GET Response .....	87
154	3.10.3	PUT Request.....	87
155	3.10.4	PUT Response .....	87
156	3.10.5	DELETE Request .....	88
157	3.10.6	DELETE Response .....	88
158	3.10.7	Binding Instance Access Example.....	88
159	3.11	Report Configuration Collection .....	88
160	3.11.1	GET Request.....	89
161	3.11.2	GET Response .....	89
162	3.11.3	POST Request.....	89
163	3.11.4	POST Response .....	91
164	3.11.5	Reporting Configuration Collection Access Example .....	92
165	3.12	Report Configuration Instance .....	93
166	3.12.1	GET Request.....	93
167	3.12.2	GET Response .....	93
168	3.12.3	PUT Request.....	93
169	3.12.4	PUT Response .....	93
170	3.12.5	DELETE Request .....	93
171	3.12.6	DELETE Response .....	93
172	3.12.7	Reporting Configuration Instance Access Example .....	94
173	3.13	Notification Resource.....	94
174	3.13.1	POST Request.....	94
175	3.13.2	POST Response .....	96
176	3.13.3	Notification Resource Access Example.....	96
177	3.14	Group Collection.....	96
178	3.14.1	GET Request.....	96
179	3.14.2	GET Response .....	96
180	3.14.3	Group Collection Access Example .....	96
181	3.15	Group Endpoint Collection .....	97
182	3.15.1	GET Request.....	97
183	3.15.2	GET Response .....	97

184	3.15.3	Group Endpoint Collection Access Example .....	97
185	3.16	Group Attribute Collection .....	97
186	3.16.1	GET Request .....	98
187	3.16.2	GET Response .....	98
188	3.16.3	POST Request .....	98
189	3.16.4	POST Response .....	99
190	3.16.5	Group Attribute Collection Access Example .....	99
191	3.17	Group Attribute Instance .....	100
192	3.17.1	GET Request .....	100
193	3.17.2	GET Response .....	100
194	3.17.3	PUT Request .....	100
195	3.17.4	PUT Response .....	100
196	3.17.5	Group Attribute Instance Access Example .....	101
197	3.18	Group Command Collection .....	101
198	3.18.1	GET Request .....	101
199	3.18.2	GET Response .....	101
200	3.18.3	Group Command Collection Access Example .....	102
201	3.19	Group Command Instance .....	102
202	3.19.1	POST Request .....	102
203	3.19.2	POST Response .....	102
204	3.19.3	Group Command Instance Access Example .....	103
205	3.20	Group Default Report Configuration Instance .....	103
206	3.20.1	GET Request .....	104
207	3.20.2	GET Response .....	104
208	3.20.3	PUT Request .....	104
209	3.20.4	PUT Response .....	104
210	3.20.5	DELETE Request .....	104
211	3.20.6	DELETE Response .....	104
212	3.20.7	Group Default Report Configuration Instance Access Example .....	105
213	3.21	Group Notification .....	105
214	3.21.1	POST Request .....	105
215	3.21.2	POST Response .....	105
216	3.21.3	Group Notification Access Example .....	106
217	4	Discovery .....	107
218	4.1	Discovery Request .....	107
219	4.1.1	Discovery Request URI .....	107
220	4.1.2	Secure Discovery Access .....	108
221	4.1.3	Attribute Value Filter .....	108



222	4.1.4	Requested Encoding of Discovery Data .....	108
223	4.1.5	Multicast Addressing for Discovery .....	108
224	4.2	Discovery Response .....	109
225	4.2.1	Response Payload Encoding .....	109
226	4.2.2	Discovery Message Processing Rules .....	109
227	4.3	Resource Attributes Used in ZCLIP .....	111
228	4.3.1	Attribute Value URN Namespace .....	111
229	4.3.2	Representation of Numeric Values .....	112
230	4.3.3	CoRE Standard Resource Attributes .....	112
231	4.3.4	Zigbee Alliance-Defined Resource Attributes .....	113
232	4.4	Resource Directory (RD) .....	113
233	4.4.1	RD Interfaces .....	114
234	4.4.2	Discovery and Configuration of RD interfaces .....	114
235	4.4.3	Registration of Resources at the RD .....	118
236	4.4.4	RD Lookup .....	120
237	4.4.5	Support for Multiple Resource Directories .....	120
238	4.4.6	Security .....	121
239	4.5	Sleepy Devices .....	121
240	5	EZ-Mode Commissioning .....	122
241	6	Security .....	123
242	6.1	Security Suites .....	123
243	6.2	DTLS Security .....	124
244	6.2.1	Security Mode .....	124
245	6.2.2	Cipher Suites Selection .....	124
246	6.2.3	Curve Selection .....	125
247	6.2.4	Point Format Selection .....	125
248	6.2.5	Signature Format Selection .....	126
249	6.2.6	Operational Certificate Support .....	126
250	6.2.7	Operational Certificate Provisioning .....	126
251	6.2.8	DTLS Session Support .....	126
252	6.3	Software Updates .....	126
253	6.4	Application Level Security .....	127
254	6.4.1	Using Application Layer Security .....	127
255	6.5	Cluster Security Requirements .....	128
256	6.6	Access Control .....	129
257	6.6.1	Terminology .....	130
258	6.6.2	Device Commissioning .....	130
259	6.6.3	Token resource .....	132

260	6.6.4	Protected Resource Request .....	134
261	6.6.5	Protected Resource Response .....	134
262	7	OTA Bootload .....	136
263	8	ZCLIP Cluster Translation Rules .....	137
264	8.1	Cluster Attributes and Metadata .....	137
265	8.2	Cluster Commands .....	137
266	8.2.1	ZCL Request Command.....	137
267	8.2.2	ZCL Response Command .....	137
268	8.2.3	Payload Translation Rules .....	137
269	8.2.4	ZCL Cluster-Specific Clarifications.....	139
270	Annex A	CoAP Response Codes.....	140
271	Annex B	ZCL General Request Command Mapping .....	141
272	Annex C	Reference Device for Test.....	142
273	C.1	ZCLIP Light .....	142
274	C.2	ZCLIP Switch.....	143
275	Annex D	Examples .....	144
276	D.1	ZCLIP URI Examples .....	144
277	D.2	Attribute URI Query Examples .....	144
278	D.3	Discovery Query Examples .....	144
279	D.3.1	Query to CoRE Root .....	144
280	D.3.2	Query using the 'rt' attribute.....	145
281	D.3.3	Query using the 'if' Attribute.....	145
282	D.3.4	Query using the 'ze' Attribute.....	146
283	D.3.5	Query for Active Clusters on a Zigbee Endpoint .....	147
284	D.3.6	Query using the 'ep' Attribute .....	148
285	Annex E	Cluster Implementations .....	149
286	E.1	Groups Cluster.....	149
287	E.1.1	Add Group .....	149
288	E.1.2	View Group.....	150
289	E.1.3	Get Group Membership .....	151
290	E.1.4	Remove Group .....	152
291	E.1.5	Remove All Groups .....	152
292	E.1.6	Add Group If Identifying .....	153
293	E.1.7	Command Extensions .....	153
294	E.2	Scenes Cluster .....	154
295	E.2.1	Extension Field Set .....	154

296	Annex F	For Future Consideration .....	155
297	F.1	Operation Over IPv4 Networks .....	155
298			
299			
300			

301

302

This page is intentionally blank

303

304

## List of Figures

306	Figure 1. Network Topology Employed by ZCLIP Devices .....	33
307	Figure 2. ZCL Client/Server Model .....	34
308	Figure 3. Overview of the Access Control Method .....	130
309		
310		

311

312

This page is intentionally blank

313

## List of Tables

314		
315	Table 1. CoAP Options Required by ZCLIP .....	38
316	Table 2. CoAP Message Types .....	39
317	Table 3. CoAP Message Types for ZCL General Commands.....	41
318	Table 4. Mandatory Standard Metadata Items for ZCLIP .....	50
319	Table 5. Optional Standard Metadata Definitions for ZCLIP .....	53
320	Table 6. Optional Standard Metadata Access Format .....	54
321	Table 7. Zigbee Alliance-Defined Metadata Definitions.....	54
322	Table 8. List of Multiple Metadata Querying Use Cases .....	55
323	Table 9. Optional Standard Restrictions to Primitive Data Types.....	56
324	Table 10. ZCL to CBOR Data Type Mapping .....	59
325	Table 11. ZCLIP Responses for ZCL Status Codes .....	68
326	Table 12. Response Codes for Nonexistent Resources .....	70
327	Table 13: ZCL Entry Point Resource Access Example.....	75
328	Table 14: Endpoint Collection Access Example .....	75
329	Table 15: Endpoint Resource Collection Access Example .....	76
330	Table 16: Cluster Resource Collection Access Example .....	77
331	Table 17. Query String Keys for Multiple Attribute Transactions .....	77
332	Table 18. Description of Attribute Query Filter Operators .....	78
333	Table 19: Attribute Collection Access Example .....	81
334	Table 20: Attribute Instance Access Example.....	82
335	Table 21: Command Collection Access Example .....	83
336	Table 22: Command Instance Access Example.....	84
337	Table 23. Binding Entry Request Parameters.....	85
338	Table 24: Binding Collection Access Example.....	87
339	Table 25: Binding Instance Access Example .....	88
340	Table 26. Report Configuration Parameters .....	89
341	Table 27. Per-Attribute Report Configuration Parameters .....	90
342	Table 28: Reporting Configuration Collection Access Example.....	92
343	Table 29: Reporting Configuration Instance Access Example .....	94
344	Table 30. Parameters for Notification Payload .....	95
345	Table 31: Notification Resource Access Example .....	96
346	Table 32: Group Collection Access Example.....	97
347	Table 33: Group Endpoint Collection Access Example .....	97
348	Table 34: Group Attribute Collection Access Example .....	99
349	Table 35: Group Attribute Instance Access Example.....	101
350	Table 36: Group Command Collection Access Example .....	102

351	Table 37: Group Command Instance Access Example.....	103
352	Table 38: Group Default Report Configuration Instance Access Example .....	105
353	Table 39: Group Notification Access Example .....	106
354	Table 40: Valid Resource URI / Link Attribute Combinations .....	111
355	Table 41: Resource Directory Configuration Collection Access Example .....	116
356	Table 42: Allowable Registration Status Transitions .....	117
357	Table 43: Resource Directory Configuration Collection Access Example .....	118
358	Table 44: Pairings of DTLS Security Mode and Cipher Suite .....	125
359	Table 45: Cluster Security Requirements .....	128
360	Table 46: Client Parameters .....	131
361	Table 47: Confirmation Claim "cnf" Parameters .....	132
362	Table 48: Access Token Parameters .....	132
363	Table 49: Access Token Confirmation Claim "cnf" Parameters .....	132
364	Table 50 Environment variables .....	133
365	Table 51: Client Request Access Token Parameters .....	134
366	Table 52: Coap Response Codes .....	140
367	Table 53: ZCL General Request Mapping .....	141
368	Table 54: ZCLIP Light Endpoint 1 Clusters .....	142
369	Table 55: ZCLIP Light Endpoint 2 Clusters .....	142
370	Table 56: ZCLIP Switch Endpoint 1 Clusters .....	143
371	Table 57: ZCLIP Switch Endpoint 2 Clusters .....	143
372	Table 58: URI Examples .....	144
373	Table 59: Attribute Filtering URI Examples .....	144
374	Table 60: Add Group Command Parameters .....	149
375	Table 61: Add Group Response Parameters .....	150
376	Table 62: View Group Command Parameters .....	150
377	Table 63: View Group Response Parameters .....	150
378	Table 64: Get Group Membership Command Parameters .....	151
379	Table 65: Get Group Membership Response Parameters .....	151
380	Table 66: Remove Group Command Parameters .....	152
381	Table 67: Remove Group Response Parameters .....	152



382

383

This page is intentionally blank

384

# 1 Introduction

## 1.1 Scope

This document specifies how to use the Zigbee Cluster Library [ZCL] over an IP stack based on IPv6 technology. The approach taken is to specify a native solution for ZCL over IP (ZCLIP), leveraging proven and well-understood IP-based application concepts and technology, rather than proposing simply to encapsulate or tunnel ZCL binary messages for transport over IP.

Elements of this specification include the following:

- IP unicast/multicast addressing, transport services
  - ZCLIP message application protocol
  - Representational State Transfer (REST) model for ZCL: URI structure, access methods, payload encoding, request/response status
  - ZCL data type mapping
  - ZCL General Command equivalents
  - Binding, reporting configuration, notification
  - Resource and Service discovery
  - Application-level Security
  - Versioning
  - General rules to transform a ZCL cluster definition into the corresponding ZCLIP form.
- Specifically, it is NOT a goal to exhaustively re-specify each ZCL cluster in terms of ZCLIP.

This effort will focus on using ZCL R07 for its work.

## 1.2 Purpose

The Zigbee Cluster Library [ZCL] specifies the functionality and intercommunication of purpose-specific application components (clusters) over the Zigbee Protocol standard [ZBPRO]. There is a large amount of accumulated knowledge, defined behavior, implementation, and deployment of the existing cluster library.

There is also significant and growing market demand to adapt the application-level functionality of the cluster library to operate over IP-based networks. The most common uses targeted are:

1. Use ZCL as an application over Thread.
2. Connect existing ZCL Zigbee PRO devices to IP connected devices (cloud, phone, or tablet).

This could be accomplished in several ways. To promote a broadly supported standard, it is important for the Zigbee Alliance to define the process and goals of using the ZCL over an IP stack. Providing this IP connectivity in a standard manner agreed upon by the Zigbee Alliance is preferred over individual companies or other organizations attempting to provide this translation.

## 1.3 Conformance testing

In order to demonstrate conformance to this specification, implementations are required to follow the appropriate test case defined in the ZCLIP Base Device Behavior Test Specification [ZCLIPTEST].

422

## 1.4 Acronyms and Abbreviations

423

<b>6LoWPAN</b>	IPv6 over Low-power Wireless Personal Area Network
<b>CoAP</b>	Constrained Application Protocol
<b>CoRE</b>	Constrained RESTful Environments
<b>CBOR</b>	Concise Binary Object Representation
<b>CT</b>	Commissioning Tool
<b>DNS</b>	Domain Name System
<b>DTLS</b>	Datagram Transport Layer Security
<b>IANA</b>	Internet Assigned Numbers Authority
<b>IETF</b>	Internet Engineering Task Force
<b>IP</b>	Internet Protocol
<b>IPv4</b>	Internet Protocol version 4
<b>IPv6</b>	Internet Protocol version 6
<b>ID</b>	Identifier
<b>JSON</b>	JavaScript Object Notation
<b>LR-WPANs</b>	Low-Rate Wireless Personal Area Networks
<b>NID</b>	Namespace Identifier
<b>NSS</b>	Namespace Specific String
<b>OTA</b>	Over-The-Air (bootloading)
<b>PSK</b>	Pre-Shared Key
<b>RD</b>	CoRE Resource Directory
<b>REST</b>	Representational State Transfer
<b>RS</b>	Resource Server
<b>UDP</b>	User Datagram Protocol
<b>UID</b>	Unique Identifier
<b>URI</b>	Uniform Resource Identifier
<b>URN</b>	Uniform Resource Name
<b>ZCL</b>	Zigbee Cluster Library
<b>ZCLIP</b>	Zigbee Cluster Library over IP

## ZDO Zigbee Device Object

### 1.5 Definitions

<b>Application Cluster</b>	An application cluster generates persistent functional application transactions between client and server.
<b>Application Transaction</b>	<p>An application (or functional) transaction is a cluster command, and possible response, that is generated to perform the device's persistent function, such as attribute reporting (e.g. reporting a sensor's measured value) or actuation commands (e.g. <i>On</i>, <i>Off</i>, <i>Toggle</i>, etc.). An application transaction is not a ZDO transaction, one-time transaction, or commissioning transaction.</p> <p>The cluster that generates the application transaction is the initiator. A corresponding cluster that receives the initial message of the transaction is the target. The same cluster on multiple endpoints/nodes could be the target of an application transaction, because of multiple source bindings or bindings with a group or broadcast destination.</p>
<b>Attribute</b>	A data entity which represents a quantity or state. This data is communicated to other devices through the attribute resources.
<b>Attribute Instance</b>	An attribute that is instantiated as part of a specific cluster instance.
<b>Binding</b>	A persistent mapping of a local cluster instance to one or more corresponding remote cluster instances. A binding can be multicast or unicast.
<b>Cluster</b>	A cluster is a specification defining one or more attributes, commands, behaviors and dependencies, that supports an independent utility or application function. The term may also be used for an implementation or instance of such a specification on an endpoint.
<b>Cluster Identifier</b>	The cluster identifier is a 16-bit number that maps to (identifies) a single cluster specification. More than one cluster identifier may map to a cluster specification, each defining a different scope and purpose.
<b>Client</b>	Typically this interface sends commands that manipulate the attributes on the corresponding server cluster. A client cluster communicates with a corresponding remote server cluster with the same cluster identifier.
<b>Corresponding Cluster</b>	The opposite side of a cluster (client to a server, or server to a client).
<b>Device</b>	A specification which defines a unique device identifier and a set of mandatory and optional clusters to be implemented on a single endpoint. The term may also be used for an implementation or instance of the device specification on an endpoint.
<b>Endpoint</b>	A particular component within a device. Each Zigbee device may support many components.
<b>Initiator Cluster</b>	An initiator cluster is an application cluster that initiates cluster transactions.

<b>Node</b>	A ZCLIP node (or node) is a single testable implementation of a ZCLIP application on a single IPv6-based Thread stack, with a single network address, on a single Thread network.
<b>Server</b>	Typically this interface supports all or most of the attributes of the cluster. A server cluster communicates with a corresponding remote client cluster with the same cluster identifier.
<b>Service Discovery</b>	The ability of a device to locate services of interest.
<b>Sleepy Node</b>	An end device on the network which spends most of its time not available on the network. Typically sleepy nodes require network layer caching support.
<b>Target Cluster</b>	A target cluster is an application cluster that receives the initiated messages from an initiator cluster and could potentially respond to the initiator.

## 426 1.6 References

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

### 432 1.6.1 Zigbee Alliance Documents

433

[ZBPRO]	05-3474-22, "Zigbee Specification"
[ZCL]	07-5123-07, "Zigbee Cluster Library Specification Revision 7"
[ZCLIPTEST]	16-07023, "ZCL over IP Test Specification"
[ZCLIPOTA]	17-07023, "Over-The-Air Upgrading revised for Dotdot"

434

### 435 1.6.2 IETF Documents

436

<a href="#">[ACE]</a>	draft-ietf-ace-oauth-authz-13, "Authentication and Authorization for Constrained Environments (ACE)"
<a href="#">[AESTLS]</a>	RFC 6655, "AES-CCM Cipher Suites for Transport Layer Security (TLS)"
<a href="#">[CBOR]</a>	RFC 7049, "Concise Binary Object Representation (CBOR)"
<a href="#">[COAP]</a>	RFC 7252, "The Constrained Application Protocol (CoAP)"
<a href="#">[COAPBLK]</a>	RFC 7959, "Block-Wise Transfers in the Constrained Application Protocol (CoAP)"

---

<a href="#">[CORELINK]</a>	RFC 6690, "Constrained RESTful Environments (CoRE) Link Format"
<a href="#">[CORERD]</a>	draft-ietf-core-resource-directory-14, "CoRE Resource Directory"
<a href="#">[COSE]</a>	RFC 8152, "CBOR Object Signing and Encryption (COSE)"
<a href="#">[CWT]</a>	RFC 8392, "CBOR Web Token"
<a href="#">[CWTP]</a>	Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs): draft-ietf-ace-cwt-proof-of-possession-03
<a href="#">[DTLS]</a>	RFC 6347, "Datagram Transport Layer Security Version 1.2"
<a href="#">[ECCTLS]</a>	RFC 7251, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS"
<a href="#">[ECCTLS-2]</a>	RFC 4492, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)"
<a href="#">[ESTCOAP]</a>	draft-ietf-ace-coap-est-04, "EST over secure CoAP (EST-coaps)"
<a href="#">[IPV6]</a>	RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification"
<a href="#">[IPV6AA]</a>	RFC 4291, "IP Version 6 Addressing Architecture"
<a href="#">[IP6CMPRS]</a>	RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks"
<a href="#">[JSONLINK]</a>	draft-ietf-core-links-json10, "Representing CoRE Formats in JSON and CBOR"
<a href="#">[KEYWORD]</a>	RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels"
<a href="#">[MC6ALLOC]</a>	RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses"
<a href="#">[MC6SCOPE]</a>	RFC 7346, "IPv6 Multicast Address Scopes"
<a href="#">[NAMEHASH]</a>	RFC 6920, "Naming Things With Hashes"
<a href="#">[OAUTH]</a>	RFC 6749, "The OAuth 2.0 Authorization Framework"
<a href="#">[OATMSC]</a>	RFC 6819, "OAuth 2.0 Threat Model and Security Considerations"
<a href="#">[POPKS]</a>	RFC 7800, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)"
<a href="#">[RPKDTLS]</a>	RFC 7250, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)"
<a href="#">[TLS]</a>	RFC 5246, "The Transport Layer Security (TLS) Protocol Version 1.2"
<a href="#">[UDP]</a>	RFC 768, "User Datagram Protocol"
<a href="#">[URI]</a>	RFC 3986, "Uniform Resource Identifier (URI): Generic Syntax"
<a href="#">[URN]</a>	RFC 8141, "Uniform Resource Names (URNs)"
<a href="#">[X509CERT]</a>	RFC 5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile"
<a href="#">[X509PKI]</a>	RFC 5758, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA"

437

438 **1.6.3 IEEE Documents**

439

[802154] IEEE Standard for Local and metropolitan area networks, "IEEE Standard 802.15.4 Low-Rate Wireless Personal Area Networks (LR-WPANs) - 2006"

440

441 **1.6.4 Thread Group Documents**

442

[THREAD] Thread Group Inc., "Thread 1.1.1 Specification"

443

444 **1.6.5 Fairhair Alliance Documents**

445

[FAIRHAIR\_RM] Fairhair Resource Model Cleared Draft version 0.7 1 March 2017

446

447 **1.6.6 NIST Documents**

448

[\[NISTHASH\]](#) NIST Special Publication 800-107 Rev 1, "Recommendations for Applications Using Approved Hash Algorithms"

449

450 **1.7 Notational and Descriptive Conventions**451 **1.7.1 Identifiers**

452 The following notation is used to represent and refer to numeric identifiers.

453 &lt;cl&gt; Cluster Instance Identifier

454 &lt;aid&gt; Attribute Identifier

455	<bid>	Binding Identifier
456	<cid>	Command Identifier
457	<eid>	Endpoint Identifier
458	<gid>	Group Identifier
459	<rid>	Report Identifier
460	<URI-Host>	Represents an IPv6 address, hostname, or UID
461	<URI-Path>	Represents the relative path of a resource
462	<URI-Port>	Represents a transport-layer port number
463	<URI-Scheme>	Represents the application layer protocol used (i.e. <i>coap</i> or <i>coaps</i> )

## 1.7.2 Descriptive References to Identifier Instances

For conciseness, phrases of the form "zzz <zid>" are understood to mean "the zzz whose identifier has the value <zid>". For example, the phrase "endpoint <eid>" is understood to mean "the endpoint whose identifier has the value <eid>".

## 1.7.3 Message Payload Contents

Where notation is used to illustrate a message payload structure, the JSON-like diagnostic notation specified in [CBOR] is used. In this notation:

- A numeric value is represented as Base 10 (decimal).
- A text value is enclosed in double quotes "".
- An array of values is delimited by square brackets [].
- A map of key-value pairs is delimited by curly braces {}, and the key and value of each pairs is separated by a colon.

For example, the payload notation

```
{"a": {0: 3, 1: "Hello"}, "f": [1, 1, 2, 3, 5, 8, 13]}
```

represents a map containing two entries, with text keys "a" and "f".

The value for the "a" entry is also a map, containing two entries with integer keys 0 and 1, which have numeric value 3 and text value "Hello", respectively.

The value of the "f" entry is an array of numeric values.

## 1.7.4 Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORD].



## 2 ZCLIP Foundation

The ZCL over IP specification defines the mechanisms used to transfer ZCL application objects over IP based connections. The application objects are not defined in this document, but rather referenced from the Zigbee Cluster Library document. This specification defines the necessary steps to translate clusters specified in that document into the form used in ZCL over IP.

ZCL over Zigbee Pro is interwoven with various stack concepts. This specification provides replication of relevant aspects as a result.

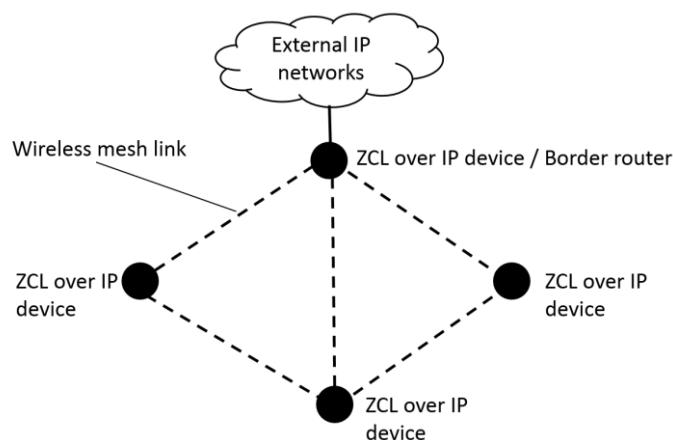
### 2.1 ZCL Overview

#### 2.1.1 System Architecture

This specification is designed to operate on a wireless mesh network based on the Thread specification [THREAD] in which devices communicate using the communication stack architecture described below.

IEEE Standard 802.15.4 [802154] is adopted to provide PHY/MAC connectivity for Low-Rate Wireless Personal Area Networks (LR-WPANs). The IPv6 header compression format for IPv6 packet delivery in Low Power Wireless Personal Area Networks (6LoWPANs) as specified in [IP6CMPRS] is adopted to provide an adaptation layer performing header compression and encapsulation for IPv6 packets to be transmitted over IEEE 802.15.4 packets. On top of the 6LoWPAN layer, IPv6 connectivity is implemented to provide addressing and routing functionalities. UDP (User Datagram Protocol) technology [UDP] is adopted as transport-layer technology.

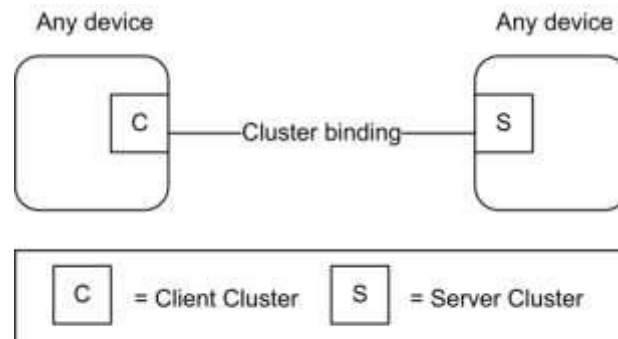
By employing the stack architecture described above, devices communicate leveraging IPv6-based connectivity over low-power low-rate wireless mesh network based on Thread specification [THREAD]. The mesh network allows full peer-to-peer communication among devices within the Thread wireless mesh. Additionally, a border router device MAY be present to provide connectivity for nodes in the mesh network to other devices in other IP-based external networks. Figure 1 illustrates the network topology employed by the devices implementing the ZCLIP functionalities described in this specification over a Thread-based network.



**Figure 1. Network Topology Employed by ZCLIP Devices**

## 2.1.2 ZCL Client/Server Model

Throughout the Zigbee Cluster Library, a client/server model is employed. This model is illustrated in Figure 2.



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

**Figure 2. ZCL Client/Server Model**

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

Requests to allow devices to manipulate attributes are (typically) sent from a device implementing the client side cluster and received by a device implementing the server side cluster. Any responses to those requests are sent to the originator of the request.

Conversely, the resource that facilitates dynamic attribute reporting is (typically) used to send from the device implementing the server side cluster (as typically this is where the attribute data itself is stored) to the device implementing the client side cluster. The report is sent to any device which has been bound to the device hosting the server side cluster.

The ZCL Client/Server roles are unrelated to the CoAP client and server roles. All certified devices will implement the ZCL client/server clusters as resources on a CoAP server. For ZCL devices that initiate commands or send reports a CoAP client implementation is used.

## 2.1.3 Endpoints

A device is composed of a collection of endpoints. Each endpoint contains a collection of application objects that compose a single logical device. Each endpoint also carries a device identifier that indicates the type of device represented by the endpoint along with a minimum set of required application objects (clusters).

Each endpoint is referenced using an endpoint identifier, represented throughout this specification by the notation <eid>. Implementers SHALL select Endpoint Identifiers between 1 and 240 unless otherwise specified by a Zigbee specification.

Sections 3.3.3 and 3.2.3 provide more details on endpoint access and operations.

## 2.1.4 Clusters

A collection of related actions and data properties is called a cluster. Each cluster focuses on a specific piece of functionality. The ZCL employs a client/server model that is described in Section 2.1.2.

Each cluster is referenced using a numeric cluster identifier. The combination of the client/server role and numeric cluster identifier is used to reference a cluster instance. Throughout this specification, this cluster instance identifier is represented by the notation <cl>. The format of <cl> follows the rules below.

- The cluster instance identifier SHALL be represented as a string.
- The cluster instance identifier SHALL begin with a 'c' for client instances and a 's' for server instances.
- For manufacturer specific extensions and clusters the Zigbee assigned manufacturer identifier SHALL be encoded as a hexadecimal value without leading zeros and appended to the string with a trailing underscore '\_'
- The numeric cluster identifier SHALL be encoded as a hexadecimal value without leading zeros and appended to the string.

Section 3.4.3 provides more details of cluster access and operations.

## 2.1.5 Attributes

Cluster related data elements that are publicly accessible through a standard mechanism are called attributes. In addition to its value, each attribute has additional metadata such as type and access permission details (read, write, report).

Each attribute is referenced using an attribute identifier, represented throughout this specification by the notation <aid>.

Sections 3.6.5 and 3.5.7 provide more details of attribute access, encoding, metadata and operations.

## 2.1.6 Commands

Cluster actions are called commands. Commands are used to execute behavior within the cluster or to query the cluster for data in more complex manners than are allowed with the attribute system.

Each command is referenced using a command identifier, represented throughout this specification by the notation <cid>.

Sections 3.8.4 and 3.7.3 provide more details of command operations.

## 2.1.7 Groups

Groups are used to execute behavior simultaneously across a number of endpoints, which can be on a single node or distributed across separate nodes. Groups are associated with multicast communication. Groups allow access to attributes and commands. Most frequently they are used in combination with the scenes cluster to trigger the recall of preset device states.

Each group is referenced using a group identifier, represented throughout this specification by the notation <gid>.

Section 3.14.3 describes the top level resource for group access.

## 2.1.8 Bindings

Bindings provide a mechanism to tie one device to a second for cluster messaging. Bindings are used by a node to send messages to one or more remote nodes when actions on a cluster are triggered. The most common use of bindings is with attribute reporting to notify remote devices of changes to attributes on a device. Bindings may also be used to generate commands to other devices.

Each binding is referenced using a binding identifier, represented throughout this specification by the notation <bid>.

Sections 3.9.5 and 3.10.7 describe the resources for binding management operations.

## 2.1.9 Reports

Reports allow a device to notify other devices in the network of changes to its attribute values. Notifications are sent to devices which have subscribed to ensure they know the current state of the attributes on the device.

Each report configuration is referenced using a report identifier, represented throughout this specification by the notation <rid>.

Sections 3.11, 3.12 and 3.20 describe the resource for report configuration management operations.

Sections 3.13 and 3.21 describe the resource for notification messages.

## 2.1.10 Manufacturer Extensions

Manufacturer extensions allow for proprietary extensions to the ZCL. Clusters specified by the ZCL may have attributes and commands added or an implementer may create a completely custom cluster. The ZCL rules for extensions and cluster creation MUST be followed. Additionally, this specification requires that all extensions SHALL be indicated at the cluster level in the URI. For extensions to an existing cluster, devices that wish to use binding functionality in extensions MUST create a binding to the extension cluster instance to receive the attribute reports and commands. This is intended to reduce the amount of unnecessary network traffic generated to devices that are unable to process the extensions.

## 2.1.11 Persistent Data

Persistent data is persistent across a restart. A restart is a program restart (warm start) or power cycle (cold start). Devices operating in the field may be restarted either manually or programmatically by maintenance personnel, or may be restarted accidentally for any number of reasons, including localized or network-wide power failures, battery replacement during the course of normal maintenance, impact, and so on. The following information SHALL be preserved across resets in order to maintain ZCLIP network operation:

- Commissioning or configuration data that allows the device to operate on the network
- Group memberships
- Bindings
- Report configurations
- Cluster attributes that represent configuration data unless specified as non-persistent

## 2.1.12 Minimal Requirements for All Devices

A node SHALL implement a binding table whose number of available entries is greater than or equal to the sum of the cluster instances, supported on each device of the node, that are initiators of application transactions.

A node SHALL have a default report configuration for every cluster that has at least one implemented attribute that is specified as mandatory and reportable.

A node that can be a target of an application transaction SHALL support group addressing and at least 8 memberships in the group table.

## 2.2 Protocol Layers

### 2.2.1 IP

#### 2.2.1.1 IP Version Support

This version of the specification SHALL use only IPv6.

#### 2.2.1.2 Address Resolution

Address resolution transforms a UID to an IP address. The methods to perform this transformation are described in Section 2.7.2.

DNS is not required in a ZCLIP network, when sending a URI to a device the requestor SHOULD NOT use hostname unless the requestor has validated that the device has access to a DNS service.

#### 2.2.1.3 Multicast Addressing

A ZCLIP Device on a normal (non-sleepy) Node SHALL support CoAP multicast access by listening on all of the IPv6 multicast scopes [MC6SCOPE] supported for the network environment in which it operates:

- Realm-Local Scope (Scope 3, ff03::xxxx). This is used to query in a 6LoWPAN based mesh network that supports Realm-Local Scope. Currently only implemented in certain IPv6 mesh network standards, e.g. Thread.
- Admin-Local Scope (Scope 4, ff04::xxxx). This is used to query over a scope that must be administratively configured, i.e. is not automatically derived from physical connectivity or other non-multicast configuration.
- Site-Local Scope (Scope 5, ff05::xxxx) This is used to query over a site-wide network. The extent of "site" is not precisely defined.

IPv6 multicasting for ZCLIP Groups is described further in Section 2.5.5.

IPv6 multicasting for ZCLIP Discovery is described further in Section 4.1.5.

### 2.2.2 UDP

This specification is designed to run over an IPv6-based Thread network. To best support resource-constrained devices and networks, the User Datagram Protocol (UDP) SHALL be the only required IP transport layer.

Both unicast and multicast transmission SHALL be supported via IP unicast and multicast addressing respectively.

### 2.2.3 CoAP

ZCLIP messages SHALL be transported between devices using the Constrained Application Protocol [CoAP].

#### 2.2.3.1 UDP Ports

The CoAP server SHALL operate on the default UDP port 5683 for unsecured URI scheme "coap" and on the default port 5684 for DTLS-secured URI scheme "coaps".

## 2.2.3.2 Message Fields

### 2.2.3.2.1 Token

The CoAP token field SHALL be used by devices to associate request and response messages. Devices MUST follow the rules specified in [CoAP]. Devices MUST provide a token of at least one octet in length, but MAY choose to use a token of greater length.

### 2.2.3.2.2 Code

The CoAP code field is used to communicate either the method for a request or the resulting status of that request in the response message. Devices MUST follow the rules for the code field specified in [COAP]. Response codes SHOULD be of the form 2.xx, 4.xx or 5.xx.

### 2.2.3.2.3 Options

The CoAP options fields are used to communicate information about a request or response as a part of the CoAP header. Devices using this specification SHALL support the following mandatory options, and may also support optional CoAP options defined in the indicated references:

CoAP Option Number	CoAP Option Name	Reference	Mandatory / Optional
3	Uri-Host	[COAP]	Optional
7	Uri-Port	[COAP]	Optional
8	Location-Path	[COAP]	Mandatory
11	Uri-Path	[COAP]	Mandatory
12	Content-Format	[COAP]	Mandatory
15	Uri-Query	[COAP]	Mandatory
17	Accept	[COAP]	Mandatory
23	Block2	[COAPBLK]	Mandatory
27	Block1	[COAPBLK]	Mandatory
28	Size2	[COAPBLK]	Optional
60	Size1	[COAP], [COAPBLK]	Optional

**Table 1. CoAP Options Required by ZCLIP**

Note that the Size1 option is first defined in [COAP] for one purpose, and [COAPBLK] extends its use for block transfers.

Devices MUST follow the rules specified in [COAP] for option processing, particularly with respect to unrecognized options. Requesting devices that receive a 4.02 Bad Option in response to a request SHOULD modify the request before attempting the request again.

### 2.2.3.3 LEISURE Multicast Response Delay

[COAP] specifies that if a response to a multicast request is to be sent, the device SHALL choose a random delay amount L between 0 and LEISURE and delay sending its response by L milliseconds.

The LEISURE parameter is a 16-bit value. The default value 0xffff represents a maximum delay of 1000 milliseconds (1 second), and the remaining values in the range 0x0000-0xfffe specify maximum delay in milliseconds (note the non-default value 0x03e8 specifies the same maximum delay as the default value).

The LEISURE is fixed at runtime but the manufacturer MAY set LEISURE to a non-default value during production. If no random delay for responses to multicast requests is desired, LEISURE parameter MAY be set to 0.

## 2.3 Transactions

The ZCL defines commands that can be sent between two devices. A command ID and a direction indicator identify the commands. For many transactions the names of the ZCL commands imply a mapping between the request and the response. The ZCL does not always explicitly call out the response to a specific command. In ZCLIP, these mappings are specified explicitly and implementations SHALL respond to a request with the specified response command. For commands without a defined response, the response SHALL be a default response with an appropriate CoAP response code and payload as specified in Section 2.3.2.

ZCL commands identified as responses may only be received in response to a request message and an implementation SHALL only process the command when received in response to a request. The ZCL response command SHALL NOT be listed in the command list resource. The ZCL response command SHALL NOT be discoverable.

### 2.3.1 CoAP Message Types for ZCL General Commands

[COAP] defines the following message types:

Message Type	Description
CReq	Confirmable CoAP Request
NReq	Non-confirmable CoAP Request
CxRsp	Confirmable Piggybacked or Separate CoAP Response
NxRsp	Non-confirmable Piggybacked or Separate CoAP Response

**Table 2. CoAP Message Types**

A confirmable message prompts an acknowledgment to be sent from the receiver of the message. The acknowledgment informs the sender that the sent message was received (and no further retries are needed).

The acknowledgment is distinct from the response message, which is sent whether or not the request was confirmable or non-confirmable.

A Piggybacked CoAP Response is bundled into the same message as the confirmable request acknowledgment. A Separate CoAP Response is sent in a separate message from the confirmable request acknowledgment, or in response to a non-confirmable request.

The ZCL General Commands map to CoAP message types as follows:

ZCL general command	CoAP Message	Method / Usage	CoAP Message Type
Read Attributes	CoAP Request	GET using query string for multiple attributes	CReq
Read Attributes Response	CoAP Response	-	CxRsp
Write Attributes	CoAP Request	PUT/POST	CReq
Write Attributes Undivided	CoAP Request	PUT/POST using query string with 'u' key	CReq
Write Attributes Response	CoAP Response	-	CxRsp
Write Attributes No Response	CoAP Request	PUT/POST	CReq
Configure Reporting	CoAP Request	POST to /zcl/e/<eid>/<cl>/r	CReq
Configure Reporting Response	CoAP Response	-	CxRsp
Read Reporting Configuration	CoAP Request	GET to /zcl/e/<eid>/<cl>/r	CReq
Read Reporting Configuration Response	CoAP Response	-	CxRsp
Report Attributes	CoAP Request	POST to ../n	CReq or NReq (multicast only)
Default Response	CoAP Response	CoAP Response Code ([COAP] section 5.9) / ZCL Error Code in payload	CxRsp
Discover Attributes	CoAP Request	GET using query string for multiple attributes	CReq
Discover Attributes Response	CoAP Response	-	CxRsp
Read Attributes Structured	-	(not supported)	(not supported)



ZCL general command	CoAP Message	Method / Usage	CoAP Message Type
Write Attributes Structured	-	(not supported)	(not supported)
Write Attributes Structured Response	-	(not supported)	(not supported)
Discover Commands Received Command	CoAP Request	GET to /zcl/e/<eid>/<cl>/c	CReq
Discover Commands Received Response	CoAP Response	-	CxRsp
Discover Commands Generated Command	-	(not supported; optional in ZCL Pro and not widely adopted)	-
Discover Commands Generated Response	-	(not supported; optional in ZCL Pro and not widely adopted)	-
Discover Attributes Extended Command	CoAP Request	See section 2.8.1.5	CReq
Discover Attributes Extended Command Response	CoAP Response	See section 2.8.1.5	CxRsp

Table 3. CoAP Message Types for ZCL General Commands

## 2.3.2 ZCL Default Response Command

In [ZCL] terminology, requests and responses are both termed "commands". A ZCL request command sent from one device to another is answered by a ZCL response command in the opposite direction. Most ZCL request commands specify a request-specific ZCL response command to be returned. When no request-specific response command is specified, or when the specified response command is not used to report errors, the Default Response command (which simply reports a ZCL Status result) is returned to the requestor. Additionally, a requestor, upon receiving a non-default response command, can answer it with a Default Response (i.e., a three-message exchange). [ZCL] provides that a ZCL command can optionally set a Disable Default Response bit in the ZCL command header to suppress the return of a Default Response reporting SUCCESS; and this setting is commonly used in non-default response commands to reduce exchanges to two messages, i.e., a request command answered by a response command.

The criteria for generating a Default Response are specified in [ZBPRO] Section 2.5.12.2 and restated here; all four of the criteria must be satisfied for a Default Response to be generated:

- A device receives a unicast command that is not a Default Response command.
- No other command is sent in response to the received command, using the same Transaction sequence number as the received command.
- The Disable Default Response bit in [the received command's] Frame Control field is set to 0...or when an error results.

- The command processing description for the received command does not override the behavior of when a Default Response command is sent.

ZCLIP largely adheres to these criteria, but with the following exceptions:

- ZCLIP messages do not contain explicit indication of Disable Default Response.
- A received ZCLIP CoAP request message, corresponding to a ZCL request command, SHALL be processed as though Disable Default Response bit IS NOT set.
- A received ZCLIP CoAP response message, corresponding to a ZCL response command, SHALL be processed as though Disable Default Response IS set.
- Moreover, ZCLIP request/response interactions are strictly two-message exchanges. If a received ZCLIP CoAP response message is found to be in error, a (third-message) Default Response SHALL NOT be sent from requestor to responder to report the error.
- The default response SHALL only apply to the command resource and it's child resources (i.e. /zcl/e/<eid>/<cl>/c, /zcl/e/<eid>/<cl>/c/<cid>, /zcl/g/<gid>/<cl>/c, and /zcl/g/<gid>/<cl>/c/<cid>).
- A default response SHALL NOT be generated if a layer other than ZCL detects the error (e.g., a non-existent command/resource, badly formatted CBOR).

See Section 2.8.2.5 for a description of ZCLIP Default Response payload.

## 2.4 Bindings

A binding is a unidirectional logical link from one device to another device. The source of a binding SHALL be a specific endpoint on a device. The destination SHALL be an endpoint or group on another device.

Reports SHALL be sent from source to destination using bindings. Devices MAY also use bindings to access attributes and commands on other nodes.

Bindings MAY be created by the source, the destination, or a third-party device. Because the source of a binding is always an endpoint, all access SHALL be through specific endpoints URIs. The group resources SHALL NOT permit access to bindings.

### 2.4.1 Binding ID

A binding is identified by an 8-bit identifier in the range 0x00 – 0xfe, encoded in CBOR as Major Type 0. The value 0xff is reserved.

NOTE: The 8-bit size is the same as the binding identifier size specified for Zigbee PRO. Further investigation would be needed to determine how a larger (16-bit) binding ID space could be used, if needed, either in a pure ZCLIP environment, or in the ZCLIP side of a mixed ZCLIP/PRO environment (where a gateway might manage bindings so as to accommodate the 8-bit constraint in the PRO network).

## 2.5 Groups

Group addressing allows for endpoints on a device to be assigned to one or more groups. Packets sent to a group will be delivered to every endpoint assigned to the group.

768 Unicast group defines a group in which all endpoints belong to the same device and packets are delivered  
769 to the endpoints through the device unicast IP address.

770 Multicast group defines a group in which endpoints MAY belong to different devices and packets are  
771 delivered to the endpoints through a multicast IP address common to all the devices which have endpoints  
772 belonging to the group.

773 Broadcast group defines a group in which all endpoints of a device belong. If multiple devices are  
774 addressable via a common multicast IP address, a broadcast group addresses all endpoints of all devices.

775 Groups cluster commands allow to add, view, delete the group membership of an endpoint.

776 Access to the resources within a group is specified via interactions with the clusters implemented at the  
777 endpoint members of a group.

## 778 **2.5.1 Group ID**

779 A group is identified by a 16-bit identifier in the range 0x0001 – 0xfff7, encoded in CBOR as Major Type  
780 0. Any endpoint on any device MAY be assigned to one or more groups identified by the corresponding  
781 group ID(s).

## 782 **2.5.2 Broadcast Group**

783 There SHALL NOT be a broadcast endpoint.

784 The group having Group ID equal to 0xffff SHALL be the Broadcast Group and SHALL contain all  
785 endpoints in a device. All devices SHALL support the Broadcast Group.

786 Requests to PUT/POST/DELETE the Broadcast Group SHOULD NOT be permitted. CoAP error code  
787 4.05 (i.e. Method Not Allowed) SHALL be returned for PUT/POST/DELETE requests to the Broadcast  
788 Group to a single device via unicast IP address. Devices receiving PUT/POST/DELETE requests to the  
789 Broadcast Group via multicast IP address SHALL ignore the request.

790 This guidance only applies to /zcl/g/ffff but not its subordinate resources.

791 See Section 2.5.5.2 for a description of how Broadcast Group ID is mapped to a multicast IPv6 address.

## 792 **2.5.3 Support for Groups Cluster**

793 The Groups Cluster provides add/view/remove management control of an endpoint's membership in  
794 groups. An endpoint capable of being added to one or more groups other than the Broadcast Group SHALL  
795 implement the Groups Cluster.

## 796 **2.5.4 No Default Response to Group Multicast**

797 Multicast messages to a group address SHALL only trigger a response if there is a cluster specific response  
798 from the targeted endpoint. Default responses SHALL NOT be returned when a multicast request is sent to  
799 a group address.

## 800 **2.5.5 Group ID Mapping to Multicast IPv6 Address**

801 Each 16-bit Group ID in the range [0x0001 – 0xfff7] corresponding to a multicast group SHALL be  
802 associated with a multicast IPv6 address at the moment of adding the group membership using the Add  
803 Group command specified in the Groups cluster (section E.1.1). The associated multicast IP address  
804 SHALL be chosen from the range of allowed addresses detailed in Section 2.5.6. The scope field of an IPv6

multicast address corresponding to a Group ID SHALL be configured according to one of the scopes listed in Section 2.2.1.3. The default scope SHALL be 5.

A ZCLIP device SHALL support automatic and manual association between Group ID and a Multicast IPv6 address. Automatic association between Group IDs and Multicast IP addresses SHALL be the default mechanism.

Automatic association SHALL map the Group ID of a ZCLIP device to the group ID field of the IPv6 multicast address specified in [IPv6AA] according to the following convention:

- The Group ID of a ZCLIP device maps to the rightmost group of four hexadecimal digits of the IPv6 multicast address, representing the rightmost 16 bits of the group ID field of the IPv6 multicast address. The mapping is obtained according to the following rule:

$$\text{GroupID\_multicast\_IPv6} = (\text{Group\_ID\_ZCLIP} - 1) \bmod (2^{\text{base}}) + 1$$

The parameter *base* in the range [0x00-0x10] represents the number of rightmost bits used to represent the range of multicast IPv6 addresses available. The recommended value of base parameter SHOULD be set to 0x0a, which allows to predictably map all possible Group IDs up to 1024 IPv6 multicast addresses. When *base* parameter equals 0x00, all possible Group IDs SHALL map to the same IPv6 multicast address. When *base* parameter equals 0x10, each distinct Group ID SHALL map to a distinct IPv6 multicast address.

- The group of four hexadecimal digits preceding the rightmost group (i.e. separated by colon (:)) in IPv6 format) SHALL contain the value 0x02cl.

Manual association SHALL specify the IPv6 multicast address associated with the Group ID of a ZCLIP device for each group as specified in the Add Group command description of the Groups cluster (Annex E.1.1).

### 2.5.5.1 Examples of IPv6 assignment

Assumption of IPv6 address possibly composed as follows (with x representing IPv6 scope hexadecimal digit):

- For a non-permanently ("dynamically") assigned multicast address, the prefix is: FF1x
- For a permanently assigned ("well-known") multicast address, the prefix is: FF0x

In both cases the prefix is followed by 112 bits of Multicast IPv6 Group ID.

Configuration	ZCL Group ID	Multicast IPv6 Address
N/A	0xffff	FF15:0:0:0:0:2c1:ffff
Automatic assignment – base=0x08	0x0005	FF1x:0:0:0:0:2c1:5
Automatic assignment – base=0x08	0x0105	FF1x:0:0:0:0:2c1:5
Automatic assignment – base=0x0a	0x0005	FF1x:0:0:0:0:2c1:5
Automatic assignment – base=0x0a	0x0105	FF1x:0:0:0:0:2c1:105
Automatic assignment – base=0x00	0x0005	FF1x:0:0:0:0:2c1:1

Automatic assignment – base=0x00	0x0105	FF1x:0:0:0:0:2c1:1
Manual assignment	0x0005	FF1x:0:0:0:0:0:1
Manual assignment	0x0105	FF1x:1234:0:0:0:0:17

838

### 839 2.5.5.2 Mapping of Broadcast Group ID to Multicast IPv6 Address

840 The Broadcast Group ID corresponding to value 0xffff SHALL be mapped to the group ID field of the IPv6  
841 multicast address specified in [IPV6AA] according to the following convention:

- 842 • The rightmost group of four hexadecimal digits of the IPv6 multicast address, representing the  
843 rightmost 16 bits of the group ID field of the IPv6 multicast address, SHALL be set to 0xffff.
- 844 • The group of four hexadecimal digits preceding the rightmost group (i.e. separated by a colon in  
845 IPv6 format) SHALL contain the value 0x02cl.
- 846 • The IPv6 multicast address used by the Broadcast Group SHALL use the scope value of 5.

## 847 2.5.6 Assignment of IPv6 Multicast Addresses

848 IPv6 multicast addresses to be used by ZCLIP multicast groups SHOULD be assigned as dynamic IPv6  
849 addresses not in the permanent range (identified by IPv6 flags as R=0, P=0, T=1) as specified in Section  
850 4.3 of [MC6ALLOC].

851 The CoAP server in a ZCLIP device normally listens to a ZCLIP group IPv6 multicast address on the  
852 default CoAP port (5683) as mandated by Section 2.2.3.1. However, to address rare cases where other  
853 CoAP/UDP applications on the same node need exclusive access to port 5683 bound to the same IPv6  
854 multicast address, a CoAP server bound to this multicast address MAY use a UDP port other than 5683.

## 855 2.6 Reporting

856 The attribute reporting mechanism provides a way for devices to configure the reports that are sent via the  
857 binding mechanism (to endpoints or groups) when an attribute value changes. Devices generating  
858 notifications SHALL support reporting for all implemented attributes marked as reportable in the ZCL and  
859 SHOULD support reporting on all other implemented attributes. This support does not imply that the  
860 attribute is included in any report configurations, but that a subscriber MAY configure reports for the  
861 attribute if desired. Devices SHALL support a single attribute reporting configuration for each cluster that  
862 contains attributes for which reporting may be configured. Devices MAY support additional attribute  
863 reporting configurations for each cluster.

864 The rules in Section 2.6.2 provide guidance on when clusters are required to support report configurations.  
865 Devices SHOULD support the creation of report configurations on any cluster which supports reporting.

### 866 2.6.1 Report ID

867 A report configuration is identified by an 8-bit identifier in the range 0x00 – 0xfe, encoded in CBOR as  
868 Major Type 0. The value 0xff SHALL indicate a null report ID. Report configurations created at runtime  
869 SHALL NOT use the report ID 0; that value is reserved for use as specified in Section 2.6.2.

### 870 2.6.2 Default Report

871 Each cluster that implements attributes specified as reportable SHALL support a default attribute reporting  
872 configuration when shipped from the factory. This will be called the default report or default reporting

configuration. The default reporting configuration SHALL contain an entry for each attribute specified as mandatory and reportable by the ZCL cluster specification. The default reporting configuration MAY also include entries for implemented attributes not specified as reportable as well. The default reporting configuration SHALL use a maximum reporting interval either of 0 or in the range 61 to 65534 seconds.

When implemented, this default report SHALL be applied to the report configuration with the ID of 0. The reporting configuration with ID of 0 MAY be overwritten at any time. In this case, the updated reporting configuration SHALL be used, and the report configuration is no longer the default reporting configuration. The default reporting configuration can be restored using the mechanisms described in Section 3.12.6.

### 2.6.3 Notification Generation

Notifications to remote devices SHALL be generated based upon the criteria contained within report configurations. A report SHALL be generated when the time that has elapsed since the previous report of the same attribute is equal to the Maximum Reporting Interval for that attribute. The time of the first report after configuration is not specified. If the Maximum Reporting Interval is set to 0, there SHALL be no periodic reporting, but change based reporting SHALL still be operational.

The device generating a notification SHALL NOT attempt to combine report configurations when generating notifications. Events that cause a change to multiple attributes (such as a command or sensed condition) SHALL NOT generate more than one notification per binding.

If a received ZCLIP request message changes the value of an attribute such that a notification would be generated, the notifier SHALL send the response to the ZCLIP request message before sending the notification.

If periodic reporting is enabled, and the maximum interval elapses, all attributes SHALL be reported in a single notification.

If a reportable change occurs in one attribute, and the minimum interval has elapsed, all attributes SHALL be reported in a single notification.

If a reportable change occurs in one attribute, and the minimum interval has not elapsed, all attributes SHALL be reported once the minimum interval elapses.

For robustness in power loss situations or similar, devices at startup SHALL wait a pseudo-random delay between *minNotificationRestart* and *maxNotificationRestart* seconds before beginning to send notifications. The value of *minNotificationRestart* equals 5 and a manufacturer MAY select a value of *maxNotificationRestart* such that *maxNotificationRestart* is greater than or equal to *minNotificationRestart*+10. Devices SHALL NOT send any reports prior to the delay expiring.

Notifications from ZCL Cluster Servers SHALL be sent to ZCL Cluster Client subscribers. Likewise, notifications from ZCL Cluster Clients SHALL be sent to ZCL Cluster Server subscribers.

### 2.6.4 Group Support

Reporting supports a limited set of group operations. Only the default report configuration on each endpoint MAY be accessed or modified through the group addressing mechanism. This is because there is no way to ensure that the creation of a report configuration across all endpoints on a device would result in a consistent group id. When multicast group operations are implemented, the problem grows. It is not possible to create bindings using group operations.



## 2.7 Unique Identifier (UID)

IPv6 addresses can change at any time. In a standard IP network, hostnames and DNS resolution services are used to maintain communication between an application client like a browser and a unique resource on the network like a web server. In the world of Zigbee EUI64 addresses served this purpose. Unfortunately, these resolution services are not readily available in the constrained world of ZCLIP.

ZCLIP requires a way for a device to both bind to another device on the network with a static unique identifier (unaided by a hostname resolution service), and verify that it is talking to an authorized device during secure communication (without the benefit of a certificate authority).

To satisfy these requirements, each device SHALL have a single UID that is associated with all of the device's IP network interfaces. The UID can be used in conjunction with resource discovery to resolve the device's current IP address on an interface. Once in possession of the IP address, a DTLS handshake in RawPublicKey or Certificate mode can be used to verify the device's identity.

### 2.7.1 UID Value Generation

A UID is associated with a public/private key pair unique to the device. This can be a RawPublicKey or a Certificate based key pair. The actual UID is a hash of the public key based on the contents defined below.

The hash function used to create the UID for ZCLIP is consistent with directives specified in [NAMEHASH] Section 2 "Hashes Are What Count" and [COAP] Section 9.1.3.2.1 "Provisioning".

Given a public key subjectPublicKeyInfo [X509CERT], the UID SHALL be the sha-256 hash of the concatenation of "zcl.uid" and the public key subjectPublicKeyInfo. The hash SHALL be stored in base64url encoding as digest value in a "named information" URI according to [NAMEHASH] Section 3, with no padding '=' characters. The authority component of the URI SHALL be empty. The digest algorithm component of the URI SHALL be "sha-256" ([NAMEHASH] Section 9.4). The resulting prefix of the UID URI SHALL, in absolute notation, be "ni:///sha-256;".

To accommodate storage constraints of binding table entries, the 256-bit hash result MAY be truncated to no fewer than the first (leftmost) 20 base64url characters, equivalent to 120 bits. Truncation SHALL occur in a multiple of 24 bits. **NOTE:** As truncation can impact the security of the application by reducing the collision resistance, as described in [NISTHASH], devices characterized by sensitive information and/or functionalities SHOULD NOT apply the truncation of the 256-bit hash. Additionally, it is recommended that the public key subjectPublicKeyInfo provide a higher collision resistance than the UID stored.

An example of this sha-256 hash algorithm output, written as an absolute UID URI:

Concatenation "zcl.uid" and **subjectPublicKeyInfo:**

```
0x7a636c2e756964046fefe2b4b1595adb73a95783f770808e2fa9726c182e2496938e7
7962d1e1605fa411156956309a23a92f664f77c169447baf05b82c91e755cf45584a466
eb95
```

'sha-256' hash:

```
0x36f6e452b1cac02a4f6a6b4ef341af8e92281f25015b081f8683a51294b56d46
```

base64url encoding:

```
'ni:///sha-256;NvbkUrHKwCpPamtO80GvjpIoHyUBWwgfho0lEpS1bUY'
```

'sha-256-120' hash ('sha-256' hash truncated to 120 bits) in base64url encoding:

```
'ni:///sha-256;NvbkUrHKwCpPamtO80Gv'
```

## 2.7.2 UID Resolution

When communicating with a host identified by a UID, a resolution from UID to IP address must be performed. When any of the following conditions have been detected, UID resolution SHALL be performed before a request to the host is attempted:

- A device detects that no valid IP address is cached for the UID.
- A previous attempt to communicate with the host failed using the IP address cached for the UID.

A device MAY choose to perform UID to IP address resolution at any time to ensure that the IP address it has for a UID is valid. Upon being configured to communicate with an address containing a UID, a device SHOULD attempt to perform UID resolution so that the IP address is known when it is needed.

UID resolution is performed using the discovery mechanisms available on the network. As such, when a resource directory is present in the network, UID resolution SHALL be performed solely via the resource directory as described in Section 2.7.2.1. In the case where a resource directory is not available on the network, UID resolution SHALL be performed using multicast discovery as described in Section 2.7.2.2.

### 2.7.2.1 Resolution using Resource Directory

To perform UID resolution using a resource directory, a device SHALL be configured with the location of the resource directory and be able to perform discovery. Resolution of a UID MAY be performed using either a partial or complete UID.

To resolve a UID, the following steps SHALL be performed:

1. A device SHALL construct a <UID Token> as follows:
  - a. For a partial UID (< 256 bits), <UID Token> = <Partial UID>\*
  - b. For a complete UID (256 bit), <UID Token> = <Full UID>
2. A device SHALL send a unicast discovery request to the resource directory with a query string: ?ep=ni:///sha-256;<UID Token>
3. The device SHALL extract the IP address from the resource directory's response and place it into the cache, marking it as untrusted.

### 2.7.2.2 Resolution using Multicast

To perform UID resolution using multicast, a device SHALL be able to perform discovery using multicast messages.

To resolve a UID, the following steps SHALL be performed:

1. A device SHALL construct a <UID Token> as follows:
  - a. For a partial UID (< 256 bits), <UID Token> = <Partial UID>\*
  - b. For a complete UID (256 bit), <UID Token> = <Full UID>
2. A device SHALL send a multicast discovery request to /.well-known/core?ep=ni:///sha-256;<UID Token>
3. The device SHALL wait for unicast responses from devices on the network. Responses are delayed for a period of up to LEISURE milliseconds. For each response received, the device SHALL extract the IP address (from the response payload or via other mechanisms) and place the address into the cache, marking it as untrusted. When a full UID is used in the query, only one response is expected. If a truncated UID is used, collisions might occur which could lead to multiple responses.



994

995 

## 2.8 ZCLIP Messaging

996 

### 2.8.1 URI Format

997 Each ZCLIP application object SHALL be served as a resource available at a Uniform Resource Identifier  
 998 (URI) that is constructed based on the type of object being accessed and the ZCL element identifiers  
 999 (endpoint ID, attribute ID, etc.) necessary to access it.

1000 Standard description of URIs is found in [URI]. The following sections specify details concerning URIs  
 1001 used to access ZCLIP resources.

1002 

#### 2.8.1.1 Entry Point

1003 The URI Path prefix `/zcl` SHALL be the entry point for access to ZCLIP resources.

1004 

#### 2.8.1.2 Representation of Numeric Values

1005 Numeric values in the URI SHALL be expressed in Base 16 (hexadecimal).

1006 Hexadecimal digits corresponding to Base 10 values 10, 11, 12, 13, 14, 15 SHALL be represented by  
 1007 lowercase characters 'a', 'b', 'c', 'd', 'e', 'f' respectively.

1008 Numeric values consisting of two or more hexadecimal digits SHALL NOT contain leading zeroes.

1009 

#### 2.8.1.3 Manufacturer Extensions

1010 Manufacturer extensions to cluster IDs SHALL be represented by prepending a manufacturer ID followed  
 1011 by an underscore, to the cluster ID.

1012 Manufacturer extensions SHALL be applied only to cluster IDs. A manufacturer extension SHALL NOT  
 1013 be applied to a sub-resource (e.g. attribute ID, command ID). Consequently, references and/or operations to  
 1014 both standard and manufacturer sub-resources cannot be combined in a single URI transaction.

1015 

#### 2.8.1.4 Query Parameters

1016 A URI MAY include a URI Query string that influences the execution behavior for the request and the  
 1017 corresponding content of the associated response.

1018 Each URI Resource description will specify the URI Query parameters, if any, pertinent to its supported  
 1019 methods.

1020 Duplicate query parameters are not allowed. If a request contains duplicate query parameters, the CoAP  
 1021 response code SHALL be 4.00 (Bad Request).

1022 

#### 2.8.1.5 Metadata

1023 Metadata is information that is descriptive of (or related to) a resource. Metadata MAY be used to  
 1024 describe, annotate, or tag any ZCLIP resource.

1025 A '\$'-sign is prepended to the name of an item to identify it as metadata. Metadata can be addressed  
 1026 individually by the following URI:

1027 `<data_resource_uri_path>/<$metadata_name >`

where `<data_resource_uri_path>` represents the resource URI path the metadata refers to and `<$metadata_name>` represents the metadata resource name itself.

Metadata SHALL be accessed via the same RESTful interface used for ZCLIP data transactions.

In order to simplify implementation, while providing enhanced interoperability with other existing eco-systems, ZCLIP leverages the standard metadata items defined by the Fairhair Alliance in [FAIRHAIR\_RM]. Mandatory and optional metadata items in the context of this specification are described in Sections 2.8.1.5.1 and 2.8.1.5.2 respectively. Additionally, Zigbee Alliance-defined metadata items MAY be optionally specified, as described in Section 2.8.1.5.3. Access to metadata items is described in Sections 2.8.1.5.4 and 2.8.1.5.5.

#### 2.8.1.5.1 Mandatory Standard Metadata Items

Table 4 describes the mandatory standard metadata items that SHALL be implemented for the indicated resource types in a device.

Metadata	Metadata Name	Type	Description	Resource Type
Base	\$base	String	Attribute data type.	<aid>

**Table 4. Mandatory Standard Metadata Items for ZCLIP**

The Base metadata SHALL contain the ZCL data type of the attribute identified by the `<data_resource_uri_path>` and specified in the "Short" column in Table 10.

#### 2.8.1.5.2 Optional Standard Metadata Items

Table 5 describes the optional standard metadata items that MAY be implemented for the indicated resource types in a device.

Metadata	Metadata name	Type	Description	Resource types
ID	\$id	String	An identifier for a resource that is not dependent on the resource's location. It SHALL be permanent and SHALL move with the resource, if it changes its location.	All
Type	\$type	String	The specific type name of the data item. For cluster resources, it coincides with the resource type 'rt' as specified in resource discovery section 4.3.3.1.	All

Metadata	Metadata name	Type	Description	Resource types
Base	\$base	String	Base data type of the resource. As specified in the "Short" column in Table 10	All except <aid>
Display Name	\$disp	String	A name for the resource intended for human consumption. It MAY be localized.	All
Description	\$desc	String	A localizable text description of the resource.	All
Unit	\$unit	Enum8 or Enum16	The engineering unit of the resource.  * <b>Informative note:</b> metadata reserved for future use to be specified.	<aid>
Minimum value	\$min	Any numeric type, either matching the data item type as defined in ZCL, or allowing coercion to the data item type.	The minimum value allowed if the resource allows for numeric representation.	<aid>
Maximum value	\$max	Any numeric type, either matching the data item type as defined in ZCL, or allowing coercion to the data item type.	The maximum value allowed if the resource allows for numeric representation.	<aid>
Access	\$acc	map16	Allowed methods to access a resource as specified in Table 6.	All
Variability	\$var	String	The variability of the value of a resource. If set to "constant", it indicates that the value of the resource is not expected to change. If set to "config", it is expected to be defined during configuration or commissioning	All

Metadata	Metadata name	Type	Description	Resource types
			and therefore it changes infrequently. If set to "control", it represents potentially continuously variable values that accept updates from a control algorithm. If set to "status", it represents potentially continuously variable values representing the live status of calculated or measured quantities.	
Volatility	\$vol	String	It expresses how values written to the associated data item will be retained. A resource value whose volatility metadata is marked "vol" (volatile) generally does not persist over device resets and power failures. A resource value marked with "nv" (nonvolatile) is intended to survive device resets and power failures. The "nvlw" (nonvolatile-limited-writes) indicates that the value is written to a form of memory that has a limited number of write cycles before wearing out, indicating to clients that this value should not be continuously changed. Additional metadata (i.e. \$max) MAY be used to notate, for example, how many write cycles are supported.	All
Limited writes	\$lw	Any numeric type	For a value written to a form of memory that has a limited number of write cycles, this metadata indicates how many write cycles are supported	All
Links	\$links		Additional relationships and links to external data.  * <b>Informative note:</b> metadata reserved for future use to be specified.	All

Metadata	Metadata name	Type	Description	Resource types
Tags	\$tags		Semantic marker tags applicable for the resource.  * <b>Informative note:</b> metadata reserved for future use to be specified.	All
Value Tags	\$vtags		Semantic value tags applicable for the resource.  * <b>Informative note:</b> metadata reserved for future use to be specified.	All
Hypertext Reference	\$href	String	The URI for the remainder of a resource value and metadata.	All

Table 5. Optional Standard Metadata Definitions for ZCLIP

Bit number	Name	Description
b0 (LSB)	Readable	This bit SHALL be set to 1 if the resource is readable using RESTful GET method. Otherwise it SHALL be set to 0.
b1	Writable	This bit SHALL be set to 1 if the resource is writable using RESTful PUT or POST method. Otherwise it SHALL be set to 0.
b2	Commandable	This bit is not applicable since it assumes a command prioritization mechanism not provided in Zigbee. This bit SHALL always be set to 0.
b3	Executable	This bit SHALL be set to 1 if a POST on the resource is allowed and results in a function call (e.g. a ZCL command). Otherwise it SHALL be set to 0.
b4	Deletable	This bit SHALL be set to 1 if the resource can be deleted by means of RESTful DELETE. Otherwise it SHALL be set to 0.
b5	Reportable	This bit SHALL be set to 1 if a change of value of the resource can be reported. Otherwise it SHALL be set to 0.
b6	Instantiable	This bit is not applicable since it is used in Fairhair Alliance specification when a child resource can be created by POST on the current resource.
b7-b15	Reserved	Reserved for future use by the Fairhair Alliance.

**Table 6. Optional Standard Metadata Access Format****2.8.1.5.3 Optional Zigbee Alliance-Defined Metadata Items**

Table 7 lists the optional metadata items that MAY be specifically defined by the Zigbee alliance for ZCLIP resources. These are identified by the metadata name which SHALL begin with a period (".") character, immediately after the "\$" sign, in accordance with the Fairhair Alliance specification for Ecosystem-defined metadata, followed by the "zcl-" prefix. Additional metadata defined by the Zigbee alliance MAY be added, following the convention specified.

Metadata	Metadata name	Type	Description	Resource types
Default value	\$.zcl-def	Matching the resource type	Default value for the resource	<aid>
Scenes Attribute Access	\$.zcl-scenes_acc	Boolean	It SHALL be set to TRUE if an attribute can accessed through a scene, given that a Scenes cluster instance is on the same endpoint. Otherwise it SHALL be set to FALSE.	<aid>

**Table 7. Zigbee Alliance-Defined Metadata Definitions****2.8.1.5.4 Access to Metadata**

To retrieve the value of <\$metadata\_name> a requesting device SHALL send a GET request to the URI <data\_resource\_uri\_path>/<\$metadata\_name>. If the requested metadata <\$metadata\_name> is supported, the device SHALL return a response with CoAP response code 2.05 Content. The response payload SHALL contain a map containing the <\$metadata\_name> as key and the <metadata\_value> of the specified metadata as value. If metadata <\$metadata\_name> is not supported, the device SHALL return a response with CoAP response code 4.04 Not Found with empty payload.

A requesting device SHOULD NOT attempt to access metadata using POST, PUT, or DELETE methods. If a request is received to access metadata using POST, PUT, or DELETE methods, a device SHALL respond with CoAP response code 4.05 Method Not Allowed.

Multiple metadata items associated with a specific resource are retrieved through a GET request to the specific resource including the query parameter *meta* described below.

The *meta* query parameter is used with the GET method to retrieve a subset of metadata items for the selected resource. The parameter has the following form:

*meta*=<\$metadata\_name>

for which <\$metadata\_name> can be either a single metadata identifier or the wildcard character \* to select all existing metadata.

- 1078 The response to a GET request with the `meta` query parameter SHALL return a map containing an entry  
 1079 with the key "<id>" for each child resource identified by its resource identifier <id>. For each key, the  
 1080 corresponding value SHALL be a map containing an entry for each metadata item matched by the `meta`  
 1081 filter where the map keys are the matched <\$metadata\_name> and the values are the corresponding  
 1082 <metadata\_value> values.
- 1083 When used together with the `meta` query parameter in a GET request to /zcl/e/<eid>/<cl>/a resource, the `f`  
 1084 query parameter specified in Section 3.5.1.1 allows to select to retrieve the metadata items of the attributes  
 1085 matched by the `f` filter.
- 1086 If a request is received which contains query parameters other than the ones specified above, the receiving  
 1087 device SHALL ignore them and consider only the query parameters described above.
- 1088 Response to a query request for multiple metadata or for a single metadata of multiple resources MAY be  
 1089 larger than the size of the payload of a single CoAP datagram. In this case, the responding device SHALL  
 1090 process the incoming message in accordance with the general rules specified in Section 2.8.3.2.
- 1091 (\*Informative note)
- 1092 Table 8 shows three use cases that illustrate the use of the query parameters specified above:

Use Case	Example Query
<b>Selected metadata item of selected attributes</b>	GET /zcl/e/<eid>/<cl>/a?meta=\$base&f=1-2 Response Code: 2.05 Payload: {1: {"\$base": <base_value>}, 2: {"\$base": <base_value>}}
<b>All metadata items of selected attributes</b>	GET /zcl/e/<eid>/<cl>/a/<aid>?meta=* Response Code: 2.05 Payload: {"\$id": <id>, "\$base": <base_value>, "\$type": <type_value>,...}
<b>All metadata items of all attributes.</b>	GET /zcl/e/<eid>/<cl>/a?meta=*&f=* Response Code: 2.05 Payload: {0: {"\$id": <id_value>, "\$type": <type_value>,...}, 1: {"\$id": <id_value>, "\$type": <type_value>,...},...}

**Table 8. List of Multiple Metadata Querying Use Cases**

#### 2.8.1.5.5 Access to Metadata in a Group

- 1096 Retrieval of metadata across multiple endpoint members of a group SHALL be supported.
- 1097 To retrieve metadata of a resource across multiple endpoints in a group identified by the group ID <gid>, a  
 1098 requesting device SHALL use the methods to access metadata specified in Section 2.8.1.5.4 but using (in  
 1099 the <data\_resource\_uri\_path> string) the group endpoint collection URI path /zcl/g/<gid> instead of the  
 1100 endpoint URI path /zcl/e/<eid>.
- 1101 The response message SHALL be as specified for common message processing rules in Section 2.8.6.
- 1102 If metadata for at least one endpoint of the group is retrieved successfully, the responding device SHALL  
 1103 construct a response with CoAP response code 2.05 Content and payload containing a map having one  
 1104 entry for each of the device's endpoints for which the metadata is successfully retrieved. Each entry has the  
 1105 endpoint identifier as the map key and the payload as defined in section 2.8.1.5.4 as map value.

### 2.8.1.5.6 Optional Applicable Standard Restrictions to Data Types

The following metadata represent restrictions to primitive data types defined by the Fairhair Alliance in [FAIRHAIR\_RM] that MAY be applicable to Zigbee resources.

Restriction	Mnemonics	Type	Description	Applies To	Reference Resource Types
Length	\$length	Unsigned Integer	Defines the maximum number of octets allowed to represent the resource.	String Binary	All

**Table 9. Optional Standard Restrictions to Primitive Data Types**

## 2.8.2 Payload Data Representation

### 2.8.2.1 Encoding Format

All devices SHALL support encoding all messages using Concise Binary Object Representation [CBOR].

In the absence of the CoAP Accept and Content-Format options in a CoAP request or in a successful CoAP response, devices SHALL assume a value of ‘application/cbor’ for each of those options. If the Content-Format is not specified for an error response containing a non-empty payload, the client SHALL assume the value is encoded as per section 5.5.2 of [COAP] (UTF-8).

Devices MAY support other encoding formats. Requests from devices that support multiple encoding formats SHOULD include a CoAP Accept option indicating the content type preferred by the device. If the server does not support the preferred format, it responds with 4.06 Not Acceptable, in which case the client SHOULD retry with any alternate format that it supports. A device MAY send a request with a payload indicated by the Content-Format option to be other than ‘application/cbor’. If the server does not support the indicated payload format, it responds with 4.15 Unsupported Content-Format, in which case the client SHOULD retry the request with a different format.

Devices that support a JSON encoding SHALL use the rules provided in Section 4 of [CBOR]. Numeric keys SHALL be converted to decimal representation in UTF-8 strings.

Devices SHALL NOT support indefinite types as specified by [CBOR]: i.e. indefinite map, indefinite array, indefinite byte string, and indefinite text string.

### 2.8.2.2 ZCL-to-CBOR Type Mapping

Table 10 defines the mapping of ZCL data type (as specified in [ZCL] 2.6.2) to CBOR data type.

ZCL data types NOT supported by ZCLIP are: Array, Structure, Set, Bag.

Class	ZCL Data Type	Short	CBOR Data Type
Null	No data	nodata	Major type 7, additional info 22 (null)
General Data	8-bit data	data8	Major type 0



Class	ZCL Data Type	Short	CBOR Data Type
Discrete	16-bit data	data16	Major type 0
	24-bit data	data24	Major type 0
	32-bit data	data32	Major type 0
	40-bit data	data40	Major type 0
	48-bit data	data48	Major type 0 (unsigned integer)
	56-bit data	data56	Major type 0 (unsigned integer)
	64-bit data	data64	Major type 0 (unsigned integer)
Logical Discrete	Boolean	bool	Major type 7, additional info 20 (false) and additional info 21 (true)
Bitmap Discrete	8-bit	map8	Major type 2 (byte string)
	16-bit	map16	Major type 2 (byte string)
	24-bit	map24	Major type 2 (byte string)
	32-bit	map32	Major type 2 (byte string)
	40-bit	map40	Major type 2 (byte string)
	48-bit	map48	Major type 2 (byte string)
	56-bit	map56	Major type 2 (byte string)
	64-bit	map64	Major type 2 (byte string)
Unsigned Integer Analog	Unsigned 8-bit integer	uint8	Major type 0 (unsigned integer)
	Unsigned 16-bit integer	uint16	Major type 0 (unsigned integer)
	Unsigned 24-bit integer	uint24	Major type 0 (unsigned integer)
	Unsigned 32-bit integer	uint32	Major type 0 (unsigned integer)
	Unsigned 40-bit integer	uint40	Major type 0 (unsigned integer)
	Unsigned 48-bit integer	uint48	Major type 0 (unsigned integer)
	Unsigned 56-bit integer	uint56	Major type 0 (unsigned integer)
	Unsigned 64-bit integer	uint64	Major type 0 (unsigned integer)
Signed Integer Analog	Signed 8-bit integer	int8	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 16-bit integer	int16	Major type 0 (unsigned integer) and Major type 1 (negative integer)

Class	ZCL Data Type	Short	CBOR Data Type
	Signed 24-bit integer	int24	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 32-bit integer	int32	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 40-bit integer	int40	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 48-bit integer	int48	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 56-bit integer	int56	Major type 0 (unsigned integer) and Major type 1 (negative integer)
	Signed 64-bit integer	int64	Major type 0 (unsigned integer) and Major type 1 (negative integer)
Enumeration Discrete	8-bit enumeration	enum8	Major type 0 (unsigned integer)
	16-bit enumeration	enum16	Major type 0 (unsigned integer)
Floating Point Analog	Semi-precision	semi	Major type 7, additional info 25 (half precision)
	Single precision	single	Major type 7, additional info 26 (single precision)
	Double precision	double	Major type 7, additional info 27 (double precision)
String Discrete	Octet string	octstr	Major type 2 (byte string)
	Character string	string	Major type 3 (text string)
	Long octet string	octstr16	Major type 2 (byte string)
	Long character string	string16	Major type 3 (text string)
Ordered Sequence Discrete	Array	array	NOT SUPPORTED
	Structure	struct	NOT SUPPORTED
Collection Discrete	Set	set	NOT SUPPORTED
	Bag	bag	NOT SUPPORTED
Time Analog	Time of day	ToD	Major type 2 (byte string)
	Date	date	Major type 2 (byte string)
	UTCTime	UTC	Major type 0 (unsigned integer)
Identifier	Cluster ID	clusterId	Major type 0 (unsigned integer)

Class	ZCL Data Type	Short	CBOR Data Type
Discrete	Attribute ID	attribId	Major type 0 (unsigned integer)
	BACnet OID	bacOID	Major type 0 (unsigned integer)
Miscellaneous	IEEE address	EUI64	Major type 2 (byte string)
Discrete	128-bit security key	key128	Major type 2 (byte string)
	Opaque	opaque	Major type 2 (byte string)
Unknown	Unknown	unk	RESERVED

**Table 10. ZCL to CBOR Data Type Mapping**

### 2.8.2.3 Arrays

Certain payload content may be structured as an array.

For any array instance, the elements of the array SHALL NOT be complex types (e.g., array or map).

For any array instance, the elements of the array SHALL be of uniform data type.

Array elements SHALL be sorted in ascending numerical or lexical order dependent on the type of the element.

### 2.8.2.4 Maps

Certain payload content may be structured as a map.

The data type of a map key SHALL be either integer or text string.

A text string map key SHALL NOT consist solely of the characters '0'-'9' (i.e. decimal digits). This is to avoid interworking incompatibilities with JSON and other technologies that cannot distinguish between numeric and text representations of decimal numbers ([CBOR] 3.7).

No ordering requirement is specified for map entries.

### 2.8.2.5 Default Response Payload

When a Default Response is specified to be returned, the payload of the response message SHALL be a map containing one entry. The key of the entry SHALL be a single character string of value "s", and the value of the entry SHALL be a ZCL Status code from [ZCL] in Section 2.8.4 indicating failure status as appropriate. The format of the ZCL status code in the default response SHALL be an unsigned integer, Major type 0.

As a Default Response payload indicates an error was encountered at the ZCL layer, the CoAP response code SHALL be in the 2.xx, 4.xx, or 5.xx ranges for all default response messages.

## 2.8.3 Payload Data Size

### 2.8.3.1 Request Payload Size

A request may be larger than the size of the payload of a single CoAP datagram. In these cases, a CoAP Block1 option SHALL be used to signal the desire for a block based transfer of the request payload as described in [COAPBLK].

When a device generating a request wishes to limit the size of its request, it SHALL use the CoAP Block1 option to specify the maximum size in its initial multicast or unicast request. The server device SHALL support Block mode transfers.

When processing a request message, a CoAP server device SHALL check if the CoAP Block1 option is present. If the option is present and the device supports the option, the device SHOULD use block-wise transfer as specified in [COAPBLK] to retrieve the full request; or, the device MAY discontinue the block-wise transfer. If the Block1 option is present, and the device does not complete the block-wise transfer, the device SHALL NOT use the partial request set received.

### 2.8.3.2 Response Payload Size

The response to a request might be larger than the size of the payload of a single CoAP datagram or the memory available on the receiving device. In these cases, a CoAP Block2 option SHALL be used to signal the desire for a block based transfer as described in [COAPBLK].

When a device generating a request wishes to limit the size of the response, it SHALL use the CoAP Block2 option to specify the maximum size in its initial multicast or unicast request. The server device SHALL support Block mode transfers.

When processing a response message, a CoAP client device SHALL check if the CoAP Block2 option is present. If the option is present and the device supports the option, the device SHOULD use block-wise transfer as specified in [COAPBLK] to retrieve the full result set for the request; or, the device MAY discontinue the block-wise transfer. If the Block2 option is present, and the device does not complete the block-wise transfer, the device SHALL NOT use the partial result set received.

## 2.8.4 ZCL Status Codes

Table 11. ZCLIP Responses for ZCL Status Codes, below enumerates the ZCL Status Codes defined in [ZCL] and specifies the response expected when an error occurs at a specific resource. The ZCL status code is returned if a Default Response is included in the response payload, in response to attribute operations, or when a specific command response indicates the field should use a ZCL status code..

Some ZCL status codes are no longer permissible as they are now detected and handled by the CoAP layers of the application. In these cases, the appropriate CoAP response code is indicated. In some cases the error code is general enough to occur in either the CoAP or ZCL layer. These cases will describe the appropriate handling. Devices SHALL use the guidance of the following table to return information about exceptions.

Enumerated Status	Value	Description	Layer	Notes
SUCCESS	0x00	Operation was successful.	CoAP	The SUCCESS status code is not used in Default Response messages in this specification. For Default Response messages, it is assumed that unless a ZCL status code is returned, the operation was successful.
FAILURE	0x01	Operation was not successful.	ZCL	
NOT_AUTHORIZED	0x7e	The sender of the command does not have authorization to carry out this command.	CoAP or ZCL	<p>Attempts to write directly to a resource requiring authentication from an unauthenticated client MUST be handled / indicated at the CoAP layer.</p> <p>The NOT_AUTHORIZED status code MAY only be returned when an attempt is made to modify a resource requiring authentication through a list.</p> <p>If this status is generated due to a lack of authentication, a response code of 4.01 SHALL be returned. If a requestor has authenticated but is still not permitted to perform the action, see ACTION_DENIED.</p>
RESERVED_FIELD_NOT_ZERO	0x7f	A reserved field/subfield/bit contains a non-zero value.	ZCL	

Enumerated Status	Value	Description	Layer	Notes
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.	ZCL	Issues with the CBOR encoding SHALL be handled at the CoAP layer. This status code only applies once the command payload reaches the ZCL layer.
UNSUP_CLUSTER_COMMAND	0x81	The specified cluster command is not supported on the device. Command not carried out.	CoAP	The UNSUP_CLUSTER_COMMAND status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.
UNSUP_GENERAL_COMMAND	0x82	The specified general ZCL command is not supported on the device.	CoAP	The UNSUP_GENERAL_COMMAND status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.

Enumerated Status	Value	Description	Layer	Notes
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.	CoAP	The UNSUP_MANUF_CLUSTER_COMMAND status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.
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.	CoAP	The UNSUP_MANUF_GENERAL_COMMAND status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.
INVALID_FIELD	0x85	At least one field of the command contains an incorrect value, according to the specification the device is implemented to.	ZCL	

Enumerated Status	Value	Description	Layer	Notes
UNSUPPORTED_ATTRIBUTE	0x86	The specified attribute does not exist on the device.	CoAP	The UNSUPPORTED_ATTRIBUTE status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.
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.	ZCL	
READ_ONLY	0x88	Attempt to write a read only attribute.	CoAP or ZCL	Attempts to write directly to an existing resource that is read only SHALL result in a CoAP response with a 4.05 response code.  The READ_ONLY status code MAY only be returned when an attempt is made to modify a read only resource through a list resource or when a command specifically calls out this as an appropriate status response.
INSUFFICIENT_SPACE	0x89	An operation failed due to an insufficient amount of free space available <sup>1</sup> .	ZCL	

<sup>1</sup> CCB 2092



Enumerated Status	Value	Description	Layer	Notes
DUPLICATE_EXISTS	0x8a	An attempt to create an entry in a table failed due to a duplicate entry already being present in the table.	ZCL	
NOT_FOUND	0x8b	The requested information (e.g., table entry) could not be found.	CoAP or ZCL	This status SHALL NOT be returned unless in a cluster specific command that specifies a status field.  This status code MUST NOT be used in response to a command acting upon unsupported attributes (access through <code>../a</code> and <code>../a/&lt;aid&gt;</code> ) or acting upon unsupported commands; instead a CoAP 4.04 response code is returned.
UNREPORTABLE_ATTRIBUTE	0x8c	Periodic reports cannot be issued for this attribute.	N/A	Creation of reports is done through the <code>../r</code> resource. To determine if an attribute will be reported, the created report configuration SHOULD be read.
INVALID_DATA_TYPE	0x8d	The data type given for an attribute is incorrect. Command not carried out.	ZCL	
INVALID_SELECTOR	0x8e	The selector for an attribute is incorrect.	ZCL	
WRITE_ONLY	0x8f	A request has been made to read an attribute that the requestor is not authorized to read. No action taken.	CoAP	The WRITE_ONLY status code is not used in this specification as the error MUST be detected at the CoAP level by returning a 4.05 response code.

Enumerated Status	Value	Description	Layer	Notes
INCONSISTENT_STARTUP_STATE	0x90	Setting the requested values would put the device in an inconsistent state on startup. No action taken.	ZCL	
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.	ZCL	
INCONSISTENT	0x92	The supplied values (e.g., contents of table cells) are inconsistent.	ZCL	
ACTION_DENIED	0x93	The credentials presented by the device sending the command are not sufficient to perform this action.	CoAP or ZCL	<p>Attempts to write directly to a resource by a client with insufficient permissions MUST be handled / indicated at the CoAP layer.</p> <p>When the requester is authenticated, but does not have sufficient permissions to perform the requested action, a response code of 4.03 SHALL be returned</p> <p>The NOT_AUTHORIZED status code MAY only be returned when an attempt is made to modify a resource through a list by a client with insufficient permissions.</p>
TIMEOUT	0x94	The exchange was aborted due to excessive response time.	CoAP or ZCL	If a TIMEOUT condition is detected at the CoAP layer, the device SHALL return a 5.00 response code.

Enumerated Status	Value	Description	Layer	Notes
ABORT	0x95	Failed case when a client or a server decides to abort the upgrade process.	ZCL	The ABORT status code is specific to the OTA Upgrade cluster.
INVALID_IMAGE	0x96	Invalid OTA upgrade image (ex. failed signature validation or signer information check or CRC check).	ZCL	The INVALID_IMAGE status code is specific to the OTA Upgrade cluster
WAIT_FOR_DATA	0x97	Server does not have data block available yet.	ZCL	The WAIT_FOR_DATA status code is specific to the OTA Upgrade cluster
NO_IMAGE_AVAILABLE	0x98	No OTA upgrade image available for a particular client.	ZCL	The NO_IMAGE_AVAILABLE status code is specific to the OTA Upgrade cluster
REQUIRE_MORE_IMAGE	0x99	The client still requires more OTA upgrade image files in order to successfully upgrade.	ZCL	The REQUIRE_MORE_IMAGE status code is specific to the OTA Upgrade cluster
NOTIFICATION_PENDING	0x9a	The command has been received and is being processed.	ZCL	
HARDWARE_FAILURE	0xc0	An operation was unsuccessful due to a hardware failure.	CoAP or ZCL	If a hardware failure is detected at the CoAP layer, the device SHALL return a 5.00 response code.

Enumerated Status	Value	Description	Layer	Notes
SOFTWARE_FAILURE	0xc1	An operation was unsuccessful due to a software failure.	CoAP or ZCL	If a software failure is detected at the CoAP layer, the device SHALL return a 5.00 response code.
CALIBRATION_ERROR	0xc2	An error occurred during calibration.	ZCL	
UNSUPPORTED_CLUSTER <sup>2</sup>	0xc3	The cluster is not supported	CoAP	The UNSUPPORTED_CLUSTER status code is not used in this specification as the error MUST be detected at the CoAP level and handled as a non-existent resource.

Table 11. ZCLIP Responses for ZCL Status Codes

## 2.8.5 Backward Compatible Request and Response Handling

The following sections specify support for backward compatibility, to accommodate future extensions to ZCLIP. Such extensions SHALL be designed such that a device that implements an earlier version of ZCLIP can ignore the extensions and interoperate with expected and predictable behavior.

### 2.8.5.1 Backward Compatible Handling of URI Query Parameters

A responding device SHALL recognize and process all URI Query parameters defined by the version of the ZCLIP base device specification implemented by the device.

A responding device SHALL ignore any URI Query parameters not defined by the version of the ZCLIP base device specification implemented by the device, and SHALL process a request containing unrecognized URI Query parameters as though those parameters were not present.

### 2.8.5.2 Backward Compatible Handling of Payload CBOR Map Key / Value Pairs

A device SHALL recognize and process all payload CBOR map key/value pairs defined by the version of the ZCLIP base device specification implemented by the device, or by the version of the ZCL Cluster specification implemented by the device.

A device SHALL ignore any payload CBOR map key/value pairs not defined by the version of the ZCLIP base device specification implemented by the device, or by the version of the ZCL Cluster specification implemented by the device; and SHALL process a message containing unrecognized payload CBOR map key/value pairs as though those pairs were not present.

<sup>2</sup> CCB 1485 and 1319

1212

1213

## 1214 2.8.6 Common Message Processing Rules

### 1215 2.8.6.1 Handling of Unicast Request Message

1216 A request message received via unicast SHALL be processed as if the resource does not exist (eg, return a  
1217 4.04) unless one of the following criteria are met:

- 1218 • The resource is located under /zcl/e
- 1219 • The resource is located under /zcl/t
- 1220 • The resource is a discovery message and the device is not currently registered with a resource  
1221 directory
- 1222 • The resource is a resource directory configuration message
- 1223 • The resource is located under /zcl/g/ffff (this replaces the broadcast endpoint from Zigbee Pro)
- 1224 • The resource is exactly /zcl/g

1225 A request message received via unicast IP addressing SHALL be handled as described in the following  
1226 sections.

#### 1227 2.8.6.1.1 General Request Validation

1228 Upon receipt of a request, a device SHALL apply the rules listed below. If any of the following checks fail,  
1229 no payload SHALL be returned, unless otherwise specified by the rule. The rules MAY be applied in any  
1230 order.

- 1231 1. The device SHALL check that any CoAP options included in the request are valid, recognized and  
1232 allowed to be used with the specific request method, as defined in [COAP] Sections 5.4, 5.4.1,  
1233 5.4.3, 5.4.4, 5.4.5, 5.9.2.3 and 5.10.7. If an invalid option is found from these checks, the device  
1234 SHALL respond with a 4.02 Bad Option response and processing SHALL cease. In case of 4.02  
1235 Bad Option Response, a diagnostic payload SHALL be included and it SHOULD start with a four-  
1236 character string encoding the first unrecognized Critical-option's number in hexadecimal notation.
- 1237 2. The device SHALL check if the request is for a protected resource. If so then the device SHALL  
1238 apply the access control rules as defined in sections 6.6.4 and 6.6.5.
- 1239 3. The device SHALL check for the existence of the requested resource. If the requested resource  
1240 does not exist, the device SHALL return a response code specified in Table 12 and processing  
1241 SHALL cease.

1242

1243

Method	Preferred Response	Permissible Response
GET	4.04 Not Found	--
PUT	4.04 Not Found	4.05 Method Not Allowed
POST	4.04 Not Found	4.05 Method Not Allowed

DELETE	2.02 Deleted	4.04 Not Found 4.05 Method Not Allowed
--------	--------------	-------------------------------------------

**Table 12. Response Codes for Nonexistent Resources**

4. The device SHALL check that the method requested is permitted for the specified resource. A method is permitted only if this specification explicitly defines the behavior of the method for a given resource. If the method is not permitted for the resource, the device SHALL respond with a 4.05 Method Not Allowed response and processing SHALL cease.
5. If the request contains a CoAP Accept option, the device SHALL check that the device can generate a response using the format specified by the option. If the device is unable to generate a response in the format the remote device has specified it can accept, it SHALL return a 4.06 Not Acceptable and processing SHALL cease.
6. If the request contains a payload, the device SHALL check that the Content-Format specified is supported. If the Content-Format is unsupported, the device SHALL return a 4.15 Unsupported Content-Format and processing SHALL cease.
7. If a PUT request does not contain a payload, the device SHALL return a 4.00 Bad Request and processing SHALL cease.
8. If the request contains a payload, the device SHALL ensure that the encoding of the payload is valid. If the payload is invalid (e.g., an error is thrown by the CBOR decoder), the device SHALL return a 4.00 Bad Request and processing SHALL cease.

#### **2.8.6.1.2 Request-Specific Validation and Execution**

Upon successful general request validation, the device SHALL process the request as specified for the URI and apply the following execution validation:

1. If an error is detected with the contents of the request (e.g., fields are missing or have an invalid range; invalid URI query parameter syntax; duplicate URI query parameters) the device SHALL return a 4.00 Bad Request message. The payload MAY be set to the Default Response payload with appropriate ZCL status code for the error, if the error generates from the ZCL layer.
2. If the content of the request is valid, but the CoAP server cannot process it at this time (e.g., temporary overload), the device SHALL return a 5.03 Service Unavailable.
3. If the content of the request is valid, but the CoAP server fails to process it because of an unexpected condition, the device SHALL return a 5.00 Internal Server Error.
4. If the content of the request is valid, but the ZCL server cannot process it at this time (e.g., temporary overload or unexpected condition), the device SHALL return a 2.xx CoAP status appropriate for the request method. The payload SHALL be set to the Default Response payload with appropriate ZCL status code for the error if the rules for Default Response generation are met as indicated in section 2.3.2.
5. The device SHALL check if the CoAP Block1 option is present, and if so, SHALL handle the response as described in Section 2.8.3.
6. Otherwise, the device SHALL execute the request-specific method and return a response indicating success.

### 2.8.6.1.3 Response Generation for Successful Request

When generating a URI-specific response for a successfully executed request, the device SHALL apply the following rules:

1. If the response payload would be an array of objects, but there are no objects, there SHALL be no content in the payload.
2. If the response payload would be a map, but there are no applicable key/value pairs, there SHALL be no content in the payload.
3. If the value portion of a key/value pair in a map would have no content the key SHALL NOT be included in the map.
4. If the response includes a payload the device SHALL encode the payload using the format specified by the CoAP Accept option if that option was provided by the client, otherwise using the 'application/cbor' format, and place this into the CoAP payload. The device SHALL add a CoAP Content-Format option set to the encoding type used.
5. If the request method was GET, the device SHALL set the code of the CoAP response to 2.05 Content.
6. If the request method was POST, then if the POST caused a resource to be created, the device SHALL set the code of the CoAP response to 2.01 Created and SHALL provide the path to the created resource using a series of CoAP Location-Path options; else, the device SHALL set the code of the CoAP response to 2.04 Changed (See [COAP] Section 5.8.2.)
7. If the request method was PUT, the device SHALL set the code of the CoAP response to 2.04 Changed.
8. If the request method was DELETE, the device SHALL set the code of the CoAP response to 2.02 Deleted.

The device SHALL return the CoAP response.

A quick reference table of default CoAP response codes for combinations of URI resource type and access method is available in Annex A, CoAP Response Codes.

### 2.8.6.2 Handling of Multicast Request Message

A request message received via multicast IP addressing SHALL be processed as described for a request message received via unicast IP addressing in Section 2.8.6.1, but with the following differences.

A request message received via multicast SHALL be discarded silently unless one of the following criteria are met:

- The resource is located under /zcl/g and a group multicast address is used
- The message is a discovery message and the device currently has multicast discovery enabled
- A specific resource specifically calls out support for multicast messaging

Devices SHOULD NOT send multicast requests that will be dropped as they may affect the performance of the network and will not achieve a result.

A response message SHALL NOT be sent if any of the following conditions are true:

- The code for the response is 4.xx or 5.xx indicating processing of the request failed.
- The code for the response is 2.xx indicating that processing of the request succeeded; and,
- the response has no payload
- the payload is an empty array or an empty map
- the payload is the Default Response payload

- the payload is a map of one or more entries, and the value of every entry either is empty, null, an empty map, an empty array or is the Default Response

If a response is to be sent, the device SHALL delay sending its response as described in Section 2.2.3.3.

Response messages SHALL be sent using unicast IP addressing.

### 2.8.6.3 Handling of Response Message

Upon receipt of a response to a request a device SHOULD perform the following steps:

The device SHOULD check the CoAP code to determine if the request was executed successfully on the remote device.

- A 2.xx code indicates that the device was able to successfully process the request and that the payload contains the response specified by the ZCL.
- A 4.xx code indicates that there was a problem with the request and that the device must modify it before it can be successfully processed. The payload MAY contain a ZCL Status Code that gives further information about the error. A diagnostic payload MAY be included in the response.
- A 5.xx code indicates that while the request was valid, the server could not process it at the time (e.g., there was a space constraint). The payload MAY contain a ZCL Status Code that gives further information about the error. The device MAY attempt to retry the request at a later time. A diagnostic payload MAY be included in the response.

The device SHOULD check the Content-Format option to ensure that the payload is encoded in a format that it is capable of processing before attempting to process and use the information contained in the response.

The device SHALL check if the CoAP Block2 option is present, and if so, SHALL handle the response as described in Section 2.8.3.

### 2.8.6.4 Default Response Message Cases for Request Commands

The following sections enumerate the cases for processing of a request command that results in generation of a Default Response.

#### 2.8.6.4.1 Unicast Request Command to a Single Endpoint

If a unicast request to a single endpoint is successful and no other response is defined to be returned, the responding device SHALL generate a response with CoAP response code 2.04 Changed and no content in the payload.

If a unicast command to a single endpoint is not successful because of a client error and no other response is defined to be returned, the responding device SHALL generate a response with CoAP response code 4.00 Bad Request and a Default Response payload containing a suitable ZCL status code.

If a unicast command to a single endpoint is not successful because of a server error and no other response is defined to be returned, the responding device SHALL generate a response with CoAP response code 5.00 Internal Server Error and a Default Response payload containing a suitable ZCL status code.

#### 2.8.6.4.2 Multicast Request Command to a Group

The response to a multicast request command to a group SHALL generate a response only if the command succeeds on at least one endpoint and its response is a non-Default response. The CoAP response code for the group response message SHALL match the 2.xx response code returned by the successful endpoint, and



1364 the payload SHALL be a map containing an entry for each such endpoint, having a key that is the endpoint  
1365 ID and having a value that is the successful non-Default response payload.

1366 No response is sent by a responding device for a multicast request command to a group if the command  
1367 fails on all of the responding device's endpoints, or if the command succeeds but the resulting response  
1368 payload for each successful endpoint is a Default Response payload.

## 1369 **2.8.7 TLV Extensions**

1370 Extensions to commands can be made in the ZCL where they are specified using a Tag Length Value  
1371 (TLV) mechanism. This mechanism was chosen so that devices which don't understand certain extensions  
1372 may still process data that follows the extension which is not understood. The CBOR encoding mechanism  
1373 for a map already provides a mechanism to encode a tag and a value. The length is unnecessary because the  
1374 CBOR mechanism for encoding data includes a length either implicitly or explicitly depending on the data  
1375 type.

1376 To encode extensions into a command, a device will transform the Tag to a key using the following  
1377 formula  $key = -(Tag + 1)$ . The key and associated value SHALL then be added to the command or  
1378 command response map.

1379 Devices which do not understand a Tag SHALL ignore it and process the message as if the Tag was not  
1380 present.

1381

### 3 ZCLIP Resources

This chapter specifies the ZCLIP URI resources, and the generation and handling of request and response messages for each resource. Largely, these descriptions represent the request-specific validation and execution phase of Common Message Processing Rules specified in Section 2.8.6 and its subsections, and specify the content and structure of the request and response payloads.

In the ensuing resource descriptions, it is understood that:

1. The Common Message Processing Rules apply to all resources. The details of those rules are not redundantly stated in the message handling descriptions for each resource. For example, if a GET request to a resource returns a list of objects, the description for the GET response will specify that the structure of the payload is an array; but will NOT explicitly restate that the CoAP code is 2.05 Content, nor that the format of the response payload is CBOR unless negotiated otherwise via CoAP Accept and Content-Format options; which are already specified in the common rules.
2. The payload of a request or a response is empty unless explicitly specified otherwise.
3. Only the methods explicitly described for a URI resource are supported for that resource. A device SHOULD NOT send a request with a method that is not supported for the target resource.

## 3.1 ZCL Entry Point Resources

The URI Path

/zcl

provides access to a collection of ZCL subordinate resources on the device.

### 3.1.1 GET Request

The requesting device SHALL send a GET request to retrieve the collection of ZCL subordinate resources from the responding device.

### 3.1.2 GET Response

The responding device SHALL return an array of the subordinate resources contained within it.

Each array element SHALL be a text string representing a URI Path suffix which, when appended to /zcl/, produces a URI Path that provides access to a subordinate resource available on the responding device.

Subordinate resources would typically be the endpoint collection ("e"), group collection ("g"), and token ("t") resources.

### 3.1.3 ZCL Entry Point Resources Access Example

URI	Method	Request Payload	Response	Notes
/zcl	GET		Response Code: 2.05 Content: ["e", "g", "t"]	An array of resources
/zcl	POST	Any	4.05 (Method Not allowed)	

/zcl	PUT	Any	4.05 (Method Not allowed)	
/zcl	DELETE	Any	4.05 (Method Not allowed)	

**Table 13: ZCL Entry Point Resource Access Example**

## 3.2 Endpoint Collection

The URI Path

/zcl/e

provides access to the collection of endpoints on the device.

### 3.2.1 GET Request

The requesting device SHALL send a GET request to retrieve the collection of endpoints on the responding device.

### 3.2.2 GET Response

The responding device SHALL return an array of endpoint identifiers.

### 3.2.3 Endpoint Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e	GET		Response Code: 2.05 Content: [0, 1, 2]	An Array of endpoint ids
/zcl/e	POST	Any	4.05 (Method Not allowed)	
/zcl/e	PUT	Any	4.05 (Method Not allowed)	
/zcl/e	DELETE	Any	4.05 (Method Not allowed)	

**Table 14: Endpoint Collection Access Example**

## 3.3 Endpoint Resource Collection

The URI Path

/zcl/e/<eid>

provides access to the collection of resources available for endpoint <eid>.

### 3.3.1 GET Request

The requesting device SHALL send a GET request to retrieve the collection of resources available for endpoint <eid> on the responding device.

### 3.3.2 GET Response

The responding device SHALL return an array of resources available for endpoint <eid>.

Each array element SHALL be a text string representing a URI Path suffix which, when appended to /zcl/e/<eid>/, produces a URI Path that provides access to a subordinate resource available on the responding device.

Subordinate resources would typically be client cluster instance identifiers (leading 'c' character) and server cluster instance identifiers (leading 's' character).

### 3.3.3 Endpoint Resource Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>	GET		Response Code: 2.05 Content: ["s0", "s3", "s4", "s5", "s6", "s8", "s300"]	An array of clusters on endpoint <eid>
/zcl/e/<eid>	POST	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>	DELETE	Any	4.05 (Method Not allowed)	

Table 15: Endpoint Resource Collection Access Example

## 3.4 Cluster Resource Collection

The URI Path

/zcl/e/<eid>/<cl>

provides access to the collection of subordinate resources for the cluster instance identified by <cl> on endpoint <eid>.

### 3.4.1 GET Request

The requesting device SHALL send a GET request to retrieve the collection of subordinate resources for the cluster instance <cl> on endpoint <eid> on the responding device.

### 3.4.2 GET Response

The responding device SHALL return an array of subordinate resources available for cluster <cl> on endpoint <eid>.

Each array element SHALL be a text string corresponding to a cluster subordinate resource. Each element SHALL represent a URI Path suffix which, when appended to /zcl/e/<eid>/<cl>/, produces a URI Path that provides access to a subordinate resource available for cluster instance <cl> on endpoint <eid> on the responding device.

Subordinate resources would typically be the attribute ("a"), command ("c"), binding ("b"), report configuration ("r"), and notification ("n") resources.

### 3.4.3 Cluster Resource Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>	GET		Response Code: 2.05 Content: ["a", "b", "c", "n", "r"]	An array of resources for cluster <cl> on endpoint <eid>
/zcl/e/<eid>/<cl>	POST	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>	DELETE	Any	4.05 (Method Not allowed)	

**Table 16: Cluster Resource Collection Access Example**

## 3.5 Attribute Collection

The URI Path

/zcl/e/<eid>/<cl>/a

provides access to the collection of attributes for cluster instance <cl> on endpoint <eid> of the device.

### 3.5.1 Query Parameters

Servers SHALL support the URI Query parameters listed in Table 17 to perform the equivalent transactions for reading and writing multiple attributes as specified in [ZCL].

Key	Value	Description
f	Attribute filter string.	Select subset of attributes by identifier.
u	(none)	Undivided write of multiple attributes.

**Table 17. Query String Keys for Multiple Attribute Transactions**

#### 3.5.1.1 Key f (multiple attribute selection filter)

The f query parameter represents a multiple attribute selection filter and it MAY be used with the GET method to select a subset of attributes to retrieve. The parameter has the following form:

f=<item>[,<item>,...]

for which the parameter's value is a comma-separated list of one or more items which comprises the multiple attribute selection filter.

The filter string SHALL NOT begin or end with the comma operator and two or more instances of the comma operator SHALL NOT appear consecutively in the filter string.

Each <item> is one of the following, operating according to the behavior defined in Table 18:

Name	Operator	Description
Attribute Identifier	<aid>	A single attribute identifier.
Count Specification	+	Attribute count specification of the form <start>+<count>. <start> SHALL be an attribute ID and <count> SHALL be a positive integer. From the list of attributes sorted in ascending order by ID, select the sequence of at most <count> existing attributes beginning with the attribute having the lowest ID equal to or greater than <start>.
Range Specification	-	Attribute range specification of the form <start>-<end>. <start> and <end> SHALL be attribute IDs with <start> strictly less than <end>. Select the set of existing attributes with IDs within this range, inclusive of the start and end values.
Wildcard	*	Select all existing attributes (i.e., range 0x0 – 0xffffffff)

**Table 18. Description of Attribute Query Filter Operators**

Count, range and wildcard specifications SHOULD be used when applicable to represent a filter as compactly as possible (for example, f=7-d instead of f=7,8,9,a,b,c,d).

### 3.5.1.2 Key u (multiple attribute undivided write)

This query parameter consists of the key 'u' only and is used with the POST method to write multiple attributes. This query parameter indicates that the operation SHALL be processed as an undivided operation, meaning that none of the attributes to be written is altered unless all can be modified successfully.

The use of GET with the 'u' query parameter is not allowed. For instance a GET to /zcl/e/<ep>/<cl>/a?u SHALL return a CoAP status code 4.05 (Method Not Allowed).

## 3.5.2 Attribute Value Error Checking

This section describes attribute value error checking that will be performed for endpoint or group attribute POST and PUT requests. This section will be referenced in the sections describing the handling of those requests.

For a POST or PUT request, an attribute value(s) presented in the request MUST be validated before it is applied. To validate an attribute value, the device SHALL make the error checks listed below, in the order shown. If an error condition is detected, the corresponding ZCL status code is noted for the attribute, and error checking for the attribute SHALL cease.

- 1497 • The general processing rules apply to determine if a resource exists.
  - 1498 • In the case of a POST request, if the attribute is not supported on this device, the attribute SHALL
  - 1499 be skipped for all further processing.
  - 1500 • If the attribute id in the payload does not match the attribute id in the URI (if any) the device
  - 1501 SHALL return a default response with a status of MALFORMED\_COMMAND and no further
  - 1502 processing will be performed.
  - 1503 • If the CBOR-encoded data type of the attribute in the payload is inconsistent with the ZCL data
  - 1504 type of the attribute, the status for this attribute SHALL be INVALID\_DATA\_TYPE.
  - 1505 • If the attribute is designated as read only, the status for this attribute SHALL be READ\_ONLY.
  - 1506 • If the device is not currently accepting changes to the value of the attribute, the status for this
  - 1507 attribute SHALL be NOT\_AUTHORIZED or READ\_ONLY.
  - 1508 • If the supplied value is not within the specified range of the attribute, the status for this attribute
  - 1509 SHALL be INVALID\_VALUE.
  - 1510 • If the device cannot support the supplied value, the status for this attribute SHALL be
  - 1511 INVALID\_VALUE.
- 1512 If no errors are detected for an attribute, the updated of the attribute SHALL be considered successful and
- 1513 no response status SHALL be recorded in the response.

### 1514 3.5.3 GET Request

- 1515 The GET method requests either a collection of attribute identifiers or a set of attribute identifiers and
- 1516 associated values depending on the omission/inclusion of the 'f' query parameter. To retrieve the collection
- 1517 of attribute identifiers for attributes supported by cluster instance <cl> on endpoint <eid> on the responding
- 1518 device, the requesting device SHALL send a GET request that does not include the 'f' query parameter in
- 1519 the URI.
- 1520 To retrieve one or more attribute identifiers with their associated values for attributes supported by cluster
- 1521 instance <cl> on endpoint <eid> on the responding device, the requesting device SHALL send a GET
- 1522 request that SHALL include the 'f' query parameter in the URI.

### 1523 3.5.4 GET Response

- 1524 If the request URI omits the 'f' query parameter, the responding device SHALL return an array of attribute
- 1525 identifiers for all attributes supported by cluster instance <cl> on endpoint <eid>.
- 1526 If the request URI includes the 'f' query parameter, the responding device SHALL return a map containing
- 1527 an entry for each supported attribute matched by the 'f' attribute filter. If the 'f' attribute filter does not
- 1528 match any supported attributes, an empty map SHALL be returned. For each map entry, the key is the
- 1529 attribute identifier, and the value is a map containing:
- 1530 • An entry having key "v" and value set to the attribute's value, encoded as appropriate for the
  - 1531 attribute's type, if the supported attribute's value was successfully retrieved.
  - 1532 • An entry having key "s" and value set to the ZCL Status indicating the reason for the failure, if the
  - 1533 supported attribute's value was NOT successfully retrieved.
- 1534 NOTE: Absence of an entry in the map for a particular attribute, which is defined for the cluster and which
- 1535 would otherwise be selected by the 'f' filter, signifies that the attribute is unsupported by the device.

### 3.5.5 POST Request

To configure the values of one or more attributes of cluster <cl> on endpoint <eid> on the responding device such that the configuration of each attribute can succeed or fail independent of the result for the other specified attributes, the requesting device SHALL send a POST request without the 'u' query parameter in the URI.

To configure the values of one or more attributes of cluster <cl> on endpoint <eid> on the responding device such that none of the attribute values are altered unless all would succeed (undivided write), the requesting device SHALL send a POST request that includes the 'u' query parameter in the URI.

The payload of the request SHALL be a map of key/value pairs. Each entry SHALL have a key with an attribute ID and a value specifying the new value to which the attribute is intended to be set.

### 3.5.6 POST Response

The device SHALL attempt to process each attribute specified in the payload of the request. For each attribute, the device SHALL make the error checks described in Section 3.5.2. If no error is detected for the attribute, and the 'u' query parameter IS NOT present in the request URI, the device SHALL write the supplied value to the identified attribute. The device then SHALL continue with the next attribute in the request payload until all attributes are processed.

After processing all attributes, if no error was detected for any of the attributes, and the 'u' query parameter IS present in the request URI, the device SHALL write all identified attributes with their corresponding supplied values.

The POST Response message is generated as follows:

- If all attributes in the request are successfully validated and written (regardless whether the 'u' query parameter is present in the request URI), the CoAP response SHALL have a status code 2.04 Changed and an empty payload.
- If one or more attributes fail error checking, then if the 'u' query parameter IS NOT present in the request URI, the CoAP response code SHALL be set to 2.04 Changed, otherwise it is set to 4.12 Precondition Failed. The payload SHALL contain a map of entries for the subset of attributes that fail error checking. Each entry is a key-value pair where the key is the attribute identifier of the failed attribute, and the value is a map containing an entry having key "s" and value set to the ZCL Status.

### 3.5.7 Attribute Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/a	GET		Response Code: 2.05 Content: [0, 1, 65533]	An array of attribute ids
/zcl/e/<eid>/<cl>/a?f=*	GET		Response Code: 2.05 Content: {0:{"v":<value>}, 1: {"v":<value>}, 65533: {"v":<value>}}	A CBOR map containing the key / value pairs for all attributes in cluster <cl> on endpoint <eid>



/zcl/e/<eid>/<cl>/a	POST	{0:<value>, 1:<value>}	2.04 Changed {1:{"s":0x88}}	Response with READ_ONLY for attribute 1.
/zcl/e/<eid>/<cl>/a?u	POST	{0:<value>, 1:<value>}	4.12 Precondition Failed {1:{"s":0x88}}	Response with Precondition Failed due to READ_ONLY for attribute 1.
/zcl/e/<eid>/<cl>/a	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/a	DELETE	Any	4.05 (Method Not allowed)	

Table 19: Attribute Collection Access Example

## 3.6 Attribute Instance

The URI Path

/zcl/e/<eid>/<cl>/a/<aid>

provides access to the value of attribute <aid> of cluster instance <cl> on endpoint <eid> on the device.

### 3.6.1 GET Request

To retrieve the value of attribute <aid> of cluster instance <cl> on endpoint <eid> on the responding device, the requesting device SHALL send a GET request.

### 3.6.2 GET Response

If attribute <aid> is supported, the CoAP response SHALL have a status code 2.05 Content and a payload as described in Section 3.5.4. The payload is selected for consistency between single and multiple attribute retrievals.

### 3.6.3 PUT Request

To configure the value of attribute <aid> of cluster instance <cl> on endpoint <eid> on the responding device, the requesting device SHALL send a PUT request. The request payload SHALL contain a map containing as a key the attribute Id of the attribute being set and as its value the value to which the attribute is to be set.

### 3.6.4 PUT Response

The device SHALL attempt to process the attribute specified in the request. The device SHALL validate the URI <aid> is available as a key in the request map. If the key is not available, the device SHALL note a ZCL status of NOT\_FOUND. The device SHALL make the error checks described in Section 3.5.2, noting the ZCL status if an error is detected.

If an error check fails, the device SHALL generate a CoAP response with status code 4.00 Bad Request and payload containing a map where the key is the attribute identifier of the failed attribute and the value is a map containing an entry having key "s" and value set to the ZCL Status.

1591 Otherwise, the device SHALL set the attribute to the supplied value and generate a CoAP response with  
 1592 status code set to 2.04 Changed and an empty payload.

### 1593 3.6.5 Attribute Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/a/<aid>	GET		Response Code: 2.05 Content: {<aid>:{ "v":<value>}}	A CBOR map containing the key/ value pair for the requested attribute where <value> is of the type specified for the attribute in the ZCL
/zcl/e/<eid>/<cl>/a/<aid>	POST	Any	Response Code: 4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/a/<aid>	PUT	{<aid>:<value>}	Response Code: 2.04 Changed No Content - or - 4.00 Bad Request {<aid>:{ "s":0x88}}	Response with CoAP status and if applicable the ZCL status code for failure, in this case READ_ONLY
/zcl/e/<eid>/<cl>/a/<aid>	DELETE	Any	Response Code: 4.05 (Method Not allowed)	

1594 Table 20: Attribute Instance Access Example

## 1595 3.7 Command Collection

1596 The URI Path

1597 /zcl/e/<eid>/<cl>/c

1598 provides access to the collection of commands for cluster instance <cl> on endpoint <eid> of the device.

### 1599 3.7.1 GET Request

1600 The requesting device SHALL send a GET request to retrieve the collection of commands for cluster  
 1601 instance <cl> on endpoint <eid> of the responding device.

## 3.7.2 GET Response

The responding device SHALL return an array of command identifiers for cluster instance <cl> on endpoint <eid>.

## 3.7.3 Command Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/c	GET		Response Code: 2.05 Content: [0, 1, 2]	An array of commands that can be executed for cluster <cl> on endpoint <eid>
/zcl/e/<eid>/<cl>/c	POST	Any	Response Code: 4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/c	PUT	Any	Response Code: 4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/c	DELETE	Any	Response Code: 4.05 (Method Not allowed)	

**Table 21: Command Collection Access Example**

## 3.8 Command Instance

The URI Path

/zcl/e/<eid>/<cl>/c/<cid>

provides access to command <cid> for cluster instance <cl> on endpoint <eid> of the device.

## 3.8.1 Command/Response Payload Structure

In [ZCL], the definition of a cluster command specifies zero or more command parameters sent in the request message that invokes the command, and zero or more response parameters in the response message returned for the command. Such parameters SHALL be encoded in the message payload as a map having one entry per parameter, as illustrated here:

```
{0: <value0>, 1: <value1>, 2: <value2>, ...}
```

The entry keys are ordinal integers starting from zero. Each entry value is a parameter value encoded as appropriate according to its ZCL data type.

The behavior and request/response parameters for commands vary widely over the full suite of ZCL clusters. Chapter 8 specifies rules for translating a command definition in [ZCL] into a form suitable for ZCLIP.

## 3.8.2 POST Request

The requesting device SHALL send a POST request to invoke command <cid> for cluster instance <cl> on endpoint <eid> of the responding device.

1625 If the ZCL cluster definition specifies a request command payload, the request SHALL contain a payload  
 1626 structured by applying ZCL-to-ZCLIP cluster command payload translation rules, defined in Chapter 8, to  
 1627 the cluster request command payload definition in [ZCL].

### 3.8.3 POST Response

1628 The responding device SHALL process the received command.

1630 If the ZCL cluster definition specifies a corresponding command response message, the responding device  
 1631 SHALL send a response with the appropriate CoAP code reflecting the result of the command execution. If  
 1632 the ZCL cluster definition specifies a response command payload, the response SHALL contain a payload  
 1633 structured by applying ZCL-to-ZCLIP cluster command payload translation rules, defined in Chapter 8, to  
 1634 the cluster response command payload definition in [ZCL].

1635 If command execution is successful, the CoAP response code SHALL be 2.04 Changed.

1636 If the ZCL cluster definition does not specify a corresponding response message, the responding device  
 1637 SHALL send a default response.

### 3.8.4 Command Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/c/<cid>	GET	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/c/<cid>	POST	{0: <arg0>, 1: <arg1>}	2.04 (Changed) Content: {0: <respl>} - OR - 2.04 (Changed) No Content Or 4.00 (Bad Request) Content: {0: {"s": <ZCL Error Status Code>}}	
/zcl/e/<eid>/<cl>/c/<cid>	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/c/<cid>	DELETE	Any	4.05 (Method Not allowed)	

Table 22: Command Instance Access Example

## 3.9 Binding Collection

The URI Path

/zcl/e/<eid>/<cl>/b

provides access to the collection of bindings for cluster instance <cl> on endpoint <eid> of the device.

### 3.9.1 GET Request

The requesting device SHALL send a GET request to retrieve a collection of binding identifiers from the responding device.

### 3.9.2 GET Response

The responding device SHALL return an array of binding identifiers for cluster instance <cl> on endpoint <eid>.

### 3.9.3 POST Request

NOTE: Also see Sections 3.11.3 and 3.11.4, where the creation of a report configuration record can also optionally created a related binding entry.

To create a binding entry, the requesting device SHALL send a POST request with the payload set to a map representing the binding entry. The fields in the binding entry are encoded as key/value pairs, as shown in Table 23.

Name	Key	Type	Encoding	Default
Destination URI	"u"	Character string	Major type 3	N/A
Report ID	"r"	8-bit unsigned integer	Major type 0	See 3.9.3.2

Table 23. Binding Entry Request Parameters

#### 3.9.3.1 Destination URI

The Destination URI field SHALL contain the URI of the binding destination cluster. The default representation of the destination URI SHALL contain the full URI including the host, the transport-layer port, and the relative ZCL path according to the following format: <URI-Scheme>://<URI-Host>:<URI-Port>/<URI-Path> where:

- <URI-Scheme> represents the application layer protocol used (i.e. *coap* or *coaps*).
- <URI-Host> represents the IPv6 address, hostname, or UID of the binding destination. A UID is distinguished from a hostname by its "sha-256;" prefix, with an implied base URI of "ni:///" according to [URI] (5.1.4).
- <URI-Port> represents the transport-layer port number of the binding destination.
- <URI-Path> represents the relative path of the resource (i.e. endpoint or group) containing the ZCLIP cluster binding destination.

The Destination URI entry MAY omit the <URI-Scheme> component and its trailing colon, and in this case *coap* or *coaps* scheme SHALL be used, depending on the security solution and or addressing scheme (unicast or multicast) adopted.

The Destination URI SHALL contain the <URI-Host> component. For a unicast binding, the <URI-Host> component SHOULD be a static host identifier such as a hostname or UID, and MAY be an IPv6 address, as the unicast addresses assigned to network nodes can change during network operation.

The Destination URI MAY omit the <URI-Port> component and its leading colon, and in this case standard coap port (5683) or coaps port (5684) SHALL be used, depending on the security solution adopted.

In case of unicast binding, the URI-Path of the Destination URI SHALL include the destination endpoint ID <eid>, i.e. /zcl/e/<eid>. For group binding the URI-Path of the Destination URI SHALL include the destination group ID <gid>, i.e. /zcl/g/<gid>.

### 3.9.3.2 Report ID

The Report ID field, if present, SHALL contain the identifier for the attribute report configuration entry to be used for the created binding. Bindings referencing the null report id SHALL not generate reports. If no report ID is present and the cluster supports a default report, the default report identifier SHALL be used. For clusters without a default report, the null report id SHALL be used when no report ID is present.

## 3.9.4 POST Response

If the Destination URI is not set or is improperly formatted the device SHALL respond with 4.00 Bad Request.

Otherwise, the device SHALL process the request.

The device SHALL create a temporary binding entry with the default values as specified in Table 23. The device SHALL then override the default values with any corresponding values from the request. This is the candidate binding entry.

The device SHALL search its binding table for an existing binding entry that matches the candidate binding entry. The candidate binding entry SHALL be considered a match of an existing binding entry only if the values of the candidate binding entry's Destination URI and Report ID are the same as those of the existing binding entry.

If a match exists, the device SHALL respond with a CoAP response with status 2.04 Changed, Location-Path options set to /zcl/e/<eid>/<cl>/b/<bid> where <bid> is the id of the existing binding entry, and an empty payload.

If no match exists and there is remaining capacity in the binding table, the candidate binding entry SHALL be stored persistently in the binding table. The device SHALL respond with a CoAP response with status with status 2.01 Created, Location-Path options set to /zcl/e/<eid>/<cl>/b/<bid> where <bid> is the id of the new binding entry, and an empty payload.

Otherwise, the device SHALL respond with 5.00 Internal Server Error.

## 3.9.5 Binding Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/b	GET		Response Code: 2.05 Content: [0,1,2]	Returns an array of binding identifiers
/zcl/e/<eid>/<cl>/b	POST	{ "u": " <URI-Scheme>://<URI-Host>:<URI-Port>/<URI-Path>", "r": <reportId> }	Response code 2.01 (Created) Location-path: "/zcl/e/<eid>/<cl>/b/<bid>"	Either responds with created and binding

			-OR- Response code 4.00 (Bad Request) -OR- Response code 5.00 (Internal Server Error)	URI or error code.
/zcl/e/<eid>/<cl>/b	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/b	DELETE	Any	4.05 (Method Not allowed)	

Table 24: Binding Collection Access Example

## 3.10 Binding Instance

The URI Path

/zcl/e/<eid>/<cl>/b/<bid>

provides access to binding instance <bid> for cluster <cl> on endpoint <eid> of the device.

### 3.10.1 GET Request

To retrieve a binding entry, the requesting device SHALL send a GET request.

### 3.10.2 GET Response

The responding device SHALL return a map representing the binding entry. The fields in the binding entry are encoded as key/value pairs in the map, as shown in Table 23.

### 3.10.3 PUT Request

To update a binding entry, the requesting device SHALL send a PUT request with the payload set to a map representing the binding entry. The fields in the binding entry are encoded as key/value pairs, as shown in Table 23. The device MAY omit fields with default values as stored in the binding table.

### 3.10.4 PUT Response

The device SHALL create a temporary binding entry with values initialized from the default values specified in Table 23. The device SHALL then override the temporary binding entry values with any corresponding values from the request. This is the candidate binding entry.

The device SHALL search its binding table for an existing binding entry that matches the candidate binding entry. The candidate binding entry SHALL be considered a match of an existing binding entry only if the values of the candidate binding entry's Destination URI and Report ID are the same as those of the existing binding entry.

If a match exists, the device SHALL respond with 4.00 Bad Request and a default response that includes the ZCL Status code DUPLICATE\_EXISTS. The default response SHALL be formatted according to the rules outlined in section 2.8.2.5.

1732 Otherwise, the candidate binding entry SHALL be stored persistently in the binding table, permanently  
1733 replacing the previous binding entry.

### 1734 3.10.5 DELETE Request

1735 To delete a binding entry, the requesting device SHALL send a DELETE request.

### 1736 3.10.6 DELETE Response

1737 The responding device SHALL permanently remove the binding entry.

### 1738 3.10.7 Binding Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/b/<bid>	GET		Response Code: 2.05 Content: { "u":<destination URI>, "r": <report id>}	
/zcl/e/<eid>/<cl>/b/<bid>	POST	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/b/<bid>	PUT	{ "u":<destination URI>, "r": <report id>}	2.04 (Changed) -OR- 4.00 (Bad Request) Content: { "s":0x8a}	If binding entry is a duplicate device responds with error.
/zcl/e/<eid>/<cl>/b/<bid>	DELETE		2.02 (Deleted) -OR- 4.04 (Not Found) -OR- 4.05 (Not Found)	

1739 **Table 25: Binding Instance Access Example**

## 1740 3.11 Report Configuration Collection

1741 The URI Path

1742 /zcl/e/<eid>/<cl>/r

1743 provides access to the collection of report configurations configured for cluster instance <cl> on endpoint  
1744 <eid> of the device.

1745 This resource and its subordinate resources SHALL be implemented on all clusters which support reporting  
1746 for one or more attributes. Attribute reporting details are listed in Section 2.6.



### 1747 3.11.1 GET Request

1748 To retrieve a collection of report configuration identifiers, the requesting device SHALL send a GET  
1749 request.

### 1750 3.11.2 GET Response

1751 The responding device SHALL return an array of attribute reporting configuration identifiers.

### 1752 3.11.3 POST Request

1753 To create a report configuration entry, the device SHALL send a POST request with the payload set to a  
1754 map containing a reporting configuration. The map SHALL contain the parameters listed in Table 26.

Name	Key	Type	Encoding	Default
Minimum Reporting Interval	"n"	16-bit unsigned integer	Major type 0	N/A
Maximum Reporting Interval	"x"	16-bit unsigned integer	Major type 0	N/A
Attribute(s)	"a"	Map of per-attribute configuration	Major type 5	N/A
Binding Destination URI (optional, see Section 3.11.3.4)	"u"	Character string	Major type 3	N/A

1755 **Table 26. Report Configuration Parameters**

1756 Note: This specification drops support for the direction indicator used in ZCL over Zigbee Pro. This was  
1757 done as the feature was not utilized by most applications. For applications that are interested in knowing  
1758 data about the reporting configuration of a device which is sending reports to them, they MAY query using  
1759 the report configuration identifier provided in the notification.

1760 The following sections describe the values of the fields in the parameter map.

#### 1761 3.11.3.1 Minimum Reporting Interval

1762 The value for the "n" key is the minimum reporting interval. This field is 16 bits in length and SHALL  
1763 contain the minimum interval, in seconds, between issuing reports of the specified attribute.

1764 If this value is set to 0, then there is no minimum limit, unless one is imposed by the specification of the  
1765 cluster using this reporting mechanism or by the application.

1766 The value of Minimum Reporting Interval SHALL be less than or equal to the value of Maximum  
1767 Reporting Interval.

#### 1768 3.11.3.2 Maximum Reporting Interval

1769 The value for the "x" key is the maximum reporting interval. This field is 16 bits in length and SHALL  
1770 contain the maximum interval, in seconds, between issuing reports of the specified attribute.

If the Maximum Reporting Interval is set to 0, there is no periodic reporting, but change based reporting is still operational.

NOTE: ZCL R07 is slated to specify the following resolution for CCB 2266: "Disallow setting the field to 0 without the minimum reporting interval field set to 0xffff. Return an error when this isn't the case."

### 3.11.3.3 Attribute(s)

The value for the "a" key is an attribute map. For each attribute in the report configuration, an entry SHALL be placed into this attribute map. The key SHALL be the attribute's identifier represented as an unsigned integer (major type 0). The value SHALL be a map (major type 5) with the fields specified in Table 27.

When none of the fields listed below are applicable for an attribute, the subscriber SHALL indicate a desire to receive notifications for an attribute by including an entry for the attribute, keyed by its attribute identifier and having an empty map as its value.

Name	Key	Type	Encoding	Default
Reportable Change	"r"	As per attribute referenced	As per attribute referenced	N/A
Low Threshold	"l"	As per attribute referenced	As per attribute referenced	N/A
High Threshold	"h"	As per attribute referenced	As per attribute referenced	N/A

**Table 27. Per-Attribute Report Configuration Parameters**

#### 3.11.3.3.1 Reportable Change

The Reportable Change field SHALL contain the minimum change to the attribute that will result in a notification being issued.

For attributes with 'analog' data type the field SHALL be included and have the same data type as the attribute. The sign (if any) of the reportable change field is ignored.

For attributes with the 'discrete' data type the field SHALL be omitted.

#### 3.11.3.3.2 Low Threshold

The Low Threshold field SHALL contain the low threshold for the attribute value. A transition of the attribute from a value at or above the low threshold to a value below the low threshold SHALL result in a notification being issued.

For attributes with 'analog' data type the field MAY be included. If included, the field SHALL have the same data type as the attribute.

For attributes with the 'discrete' data type the field SHALL be omitted.

#### 3.11.3.3.3 High Threshold

The High Threshold field SHALL contain the high threshold for the attribute value. A transition of the attribute from a value at or below the high threshold to a value above the high threshold SHALL result in a notification being issued.

1801 For attributes with 'analog' data type the field MAY be included. If included, the field SHALL have the  
1802 same data type as the attribute.

1803 For attributes with the 'discrete' data type the field SHALL be omitted.

#### 1804 **3.11.3.4 Binding Destination URI**

1805 The value of the "u" key is the Binding Destination URI, which is identical to the binding entry Destination  
1806 URI field specified in Section 3.9.3.1.

1807 The Binding Destination URI parameter is OPTIONAL and MAY be included only in a Report  
1808 Configuration Collection POST request. It SHALL NOT be included in a Binding Instance GET response  
1809 or PUT request.

1810 A binding entry associated with a report configuration record MAY be created together with the report  
1811 configuration record. This action SHALL have the same effect as the creation of a report configuration  
1812 record only, followed by the creation of a binding entry as specified in Section 3.9.3. The Report ID  
1813 parameter in the created binding entry SHALL be equal to the <rid> of the created reporting configuration  
1814 record as described in Section 3.11.4.

1815 To request that only a report configuration be created, the device SHALL NOT include the Binding  
1816 Destination URI in the POST request payload map.

1817 To request that a report configuration record and a binding entry be created in one transaction, the device  
1818 SHALL include the Binding Destination URI field in the POST request payload map.

### 1819 **3.11.4 POST Response**

1820 If the attribute map does not contain at least one reportable attribute ID supported by the device, the device  
1821 SHALL respond with 4.00 Bad Request with No Content.

1822 If any conditions imposed upon non attribute fields are not met the device SHALL respond with a 4.00 Bad  
1823 Request with No Content.

1824 If there is no remaining capacity in the attribute reporting table, the device SHALL respond with a 5.00  
1825 Internal Server Error with No Content.

1826 Otherwise, the device SHALL process the request. The device shall begin to build a response payload by  
1827 generating a map with the key "a" and an empty map as the value. This internal map is called the attribute  
1828 response map.

1829 For each attribute in the request, the device SHALL evaluate the request using the following rules:

1830 If the device does not implement the attribute, the device SHALL not add an entry to the attribute response  
1831 map.

1832 If any of the conditions imposed upon the attribute values are not met, the device SHALL add an entry to  
1833 the attribute response map with the attribute id as the key, and a map containing the key "s" and a value  
1834 indicating INVALID\_FIELD.

1835 The attribute reporting configuration entry SHALL be stored persistently in the attribute reporting  
1836 configuration table. The server SHALL NOT attempt to combine identical entries during the creation  
1837 process.

1838 If the device supports the Binding Destination URI parameter and that parameter is present in the map, the  
1839 device SHALL additionally attempt to create a binding entry, using the Binding Destination URI and the  
1840 identifier of the new report configuration record as the binding entry Destination URI and Report ID. The  
1841 binding entry creation attempt SHALL be processed as specified in Section 3.9.4, except that the device

SHALL only take note of the success or failure of the binding entry creation attempt, but SHALL NOT generate the corresponding response messages.

If the device supports the Binding Destination URI and that parameter is present in the request map it SHALL add an entry with the key 'u' to response payload map. If a binding was successfully created, the value SHALL be set to a CBOR-encoded text string representing the URI Path /zcl/e/<eid>/<cl>/b/<bid>, where <bid> is the identifier of the binding entry. In the case of a failure, the value SHALL be a map with key "s" and the value FAILURE.

Devices that do not support the Binding Destination URI will not add a "u" entry to the payload.

Once processing has been completed, the device SHALL perform the following steps to collapse the response.

If the attribute response map does not contain any key-value pairs, the attribute response map along with the associated key 'a' SHALL be removed from the response payload

If the response payload is an empty map, the response payload shall be set to no content

The device SHALL respond with CoAP response with status 2.01 Created and Location-Path options set to /zcl/e/<eid>/<cl>/r/<rid>, where <rid> is the id of the new attribute reporting configuration entry. A requesting device MAY request the newly created report configuration to see if any attributes were unable to be reported.

### 3.11.5 Reporting Configuration Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/r	GET		Response Code: 2.05 Content: [0, <reportID1>, <reportID2>]	An array of Attribute reporting configuratio n identifiers
/zcl/e/<eid>/<cl>/r/	POST	{"a": {0: {"r": <ReportableChange>}}, "n":<minInterval>, "x": <maxInterval>}	Response Code: 2.01 Location-Path: zcl/e/<eid>/<cl>/r /<rid>  -OR-  Response Code: 2.01 {"a": {0: {"s": 0x87}}}  -OR-  Response code: 5.00	Response where <rid> is the identifier of the reporting entry.
/zcl/e/<eid>/<cl>/r	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/r	DELETE	Any	4.05 (Method Not allowed)	

**Table 28: Reporting Configuration Collection Access Example**

## 1862 3.12 Report Configuration Instance

1863 The URI Path

1864 /zcl/e/<eid>/<cl>/r/<rid>

1865 provides access to report configuration <rid> configured for cluster instance <cl> on endpoint <eid> of the  
1866 device.

### 1867 3.12.1 GET Request

1868 To retrieve an attribute reporting configuration entry, the requesting device SHALL send a GET request.

### 1869 3.12.2 GET Response

1870 The device SHALL respond with the payload set to a map representing the reporting entry. The fields in the  
1871 reporting entry are encoded as key/value pairs, as shown in Table 26, except that the Binding Destination  
1872 URI field SHALL NOT be included.

### 1873 3.12.3 PUT Request

1874 To update an attribute configuration entry, the device SHALL send a PUT request with the payload set to a  
1875 map representing the reporting entry. The fields in the reporting entry are encoded as key/value pairs, as  
1876 shown in Table 26, except that the Binding Destination URI field SHALL NOT be included.

### 1877 3.12.4 PUT Response

1878 The device SHALL perform all error checks specified in Section 3.11.4 and then process the request as  
1879 specified in this section.

1880 If the attribute report configuration is the default report, the device SHALL apply the rules in Section 2.6.2.

1881 Otherwise, the attribute report configuration entry SHALL be stored persistently in the attribute report  
1882 configuration table for the cluster, permanently replacing the previous reporting entry.

### 1883 3.12.5 DELETE Request

1884 To delete an attribute reporting configuration entry, the device SHALL send a DELETE request.

### 1885 3.12.6 DELETE Response

1886 For entries that use the default report configuration identifier, a device SHALL remove the existing default  
1887 report configuration and immediately create a new entry in its place with the factory defaults for the report  
1888 configuration.

1889 For report configurations with an identifier that does not match the default report configuration identifier, a  
1890 device SHALL permanently remove the attribute report configuration entry. The device SHALL then  
1891 iterate through each entry in the binding table and update any entry that references deleted attribute report  
1892 configuration to point to the null report configuration.

### 3.12.7 Reporting Configuration Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/r/<rid>	GET		Response Code: 2.05 Content: { "a": {<a0>: { }}, "n":0, "x": 65534}	
/zcl/e/<eid>/<cl>/r/<rid>	POST	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/r/<rid>	PUT	{ "a": {0: { "r": <ReportableChange> } }, "n": <minInterval>, "x": <maxInterval> }	Response Code: 2.04	
/zcl/e/<eid>/<cl>/r/<rid>	DELETE		Response Code: 2.02, 4.04 or 4.05	

Table 29: Reporting Configuration Instance Access Example

## 3.13 Notification Resource

The URI Path

/zcl/e/<eid>/<cl>/n

provides access to the report notification resource for cluster instance <cl> on endpoint <eid> of the device.

### 3.13.1 POST Request

To notify subscribed endpoint <eid> of an attribute change, the device SHALL send a POST request. The payload SHALL consist of a map containing the following entries:

Name	Key	Type	Encoding	Default
Sender URI	"u"	Character string	Major type 3	N/A
Report ID	"r"	8-bit unsigned integer	Major type 0	N/A
Binding ID	"b"	8-bit unsigned integer	Major type 0	N/A
Timestamp	"t"	UTCTime	Major type 0	Not present
Attribute(s)	"a"	Map of attribute values	Major type 5	N/A

**Table 30. Parameters for Notification Payload****3.13.1.1 Sender URI**

The Sender URI entry SHALL be set to the URI of the cluster resource of the reporting device, i.e. the sender of the notification. The default representation of the URI SHALL contain the full URI including the sending host, the transport-layer port and relative ZCL path according to the following format: *<URI-Scheme>://<URI-Host>:<URI-Port>/<URI-Path>/* where:

- *<URI-Scheme>* represents the application layer protocol used (i.e. *coap* or *coaps*).
- *<URI-Host>* represents the IPv6 address, hostname, or UID of the binding destination. A UID is distinguished from a hostname by its "sha-256;" prefix, with an implied base URI of "ni:/// " according to [URI] (5.1.4). The hostname SHOULD be fully qualified. The UID SHOULD be preferred over the hostname.
- *<URI-Port>* represents the transport-layer port number of the reporting device.
- *<URI-Path>* represents the relative path of the ZCLIP cluster reporting notifications (i.e. */zcl/e/<eid>/<cl>*).

The Sender URI entry MAY omit the *<URI-Scheme>* component and its trailing colon, and in this case *coap* or *coaps* scheme SHALL be used, depending on the security solution and or addressing scheme (multicast or unicast) adopted.

The Sender URI entry MAY omit the *<URI-Host>* component and in this case the IPv6 address or hostname of the sender of the notification SHALL be used. If optional UID is used, the *<URI-Host>* component SHALL contain the UID of the sender of the notification.

The Sender URI MAY omit the *<URI-Port>* component and its leading colon, and in this case standard *coap* port (5683) or *coaps* port (5684) SHALL be used, depending on the security solution adopted.

**3.13.1.2 Report ID**

The Report ID field SHALL contain the identifier for the attribute report configuration entry that caused the notification to be generated.

**3.13.1.3 Binding ID**

The Binding ID field SHALL contain the identifier for the binding entry that caused the notification to be sent to the receiver.

**3.13.1.4 Timestamp**

The Timestamp field SHOULD be included if the reporting device has access to Universal Coordinated Time (UTC) timing functionality (whether or not the device supports the ZCL Time Cluster).

The Timestamp field SHALL be set to the timestamp of the reporting device's understanding of when the attribute change occurred. The value represents the number of seconds since 0 hours 0 minutes 0 seconds on 1 January 2000 UTC, consistent with the time standard of the ZCL Time Cluster.

**3.13.1.5 Attributes**

The Attributes field is a map of attribute value entries. For each attribute in the corresponding report configuration, there SHALL be an entry in this map with key set to the attribute ID, encoded as an unsigned integer, and the value set to the attribute's value, encoded according to the attribute type.

### 3.13.2 POST Response

If the device supports the notification resource, the device SHALL process the request and return a response with CoAP code 2.04 Changed.

### 3.13.3 Notification Resource Access Example

URI	Method	Request Payload	Response	Notes
/zcl/e/<eid>/<cl>/n	GET	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/n	POST	{ "u": <value>, "r": <value>, "b": <value>, "t": <value>, "a": <map of attribute values> }	2.04 (Changed)	
/zcl/e/<eid>/<cl>/n	PUT	Any	4.05 (Method Not allowed)	
/zcl/e/<eid>/<cl>/n	DELETE	Any	4.05 (Method Not allowed)	

Table 31: Notification Resource Access Example

## 3.14 Group Collection

The URI Path

/zcl/g

provides access to the collection of group identifiers on the device. The Broadcast Group identifier 0xffff SHALL always be present.

### 3.14.1 GET Request

To retrieve a collection of group IDs, the device SHALL send a GET request to URI-Path /zcl/g.

### 3.14.2 GET Response

The device SHALL return a payload set to an array containing one instance of each group identifier for which at least one of the device's endpoints is a member of the corresponding group.

Minimally, the returned array will contain one entry for the Broadcast Group, of which every endpoint is a member.

### 3.14.3 Group Collection Access Example

URI	Method	Request Payload	Response	Notes
-----	--------	-----------------	----------	-------



/zcl/g	GET		2.05 (Content) [1, 2, 65535]	
/zcl/g	POST	Any	4.05 (Method Not allowed)	
/zcl/g	PUT	Any	4.05 (Method Not allowed)	
/zcl/g	DELETE	Any	4.05 (Method Not allowed)	

**Table 32: Group Collection Access Example**

## 3.15 Group Endpoint Collection

The URI Path

/zcl/g/<gid>

provides access to the collection of endpoints on the device that are members of group <gid>.

### 3.15.1 GET Request

To retrieve the set of endpoints that are members of group <gid>, the device SHALL send a GET request.

### 3.15.2 GET Response

The device SHALL respond with a payload set to an array of endpoint identifiers for the endpoints on the device that are members of group <gid>.

### 3.15.3 Group Endpoint Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>	GET		2.05 (Content) [1, 2]	Array of endpoints that are members of group identifier <gid>
/zcl/g/<gid>	POST	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>	PUT	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>	DELETE	Any	4.05 (Method Not allowed)	

**Table 33: Group Endpoint Collection Access Example**

## 3.16 Group Attribute Collection

The URI Path

/zcl/g/<gid>/<cl>/a

1975 provides access to the collection of attributes for cluster instance <cl> on endpoints that are members of  
 1976 group <gid>.

### 1977 3.16.1 GET Request

1978 The GET method requests, per endpoint that is a member of group <gid>, either a collection of attribute  
 1979 identifiers or a set of attribute identifiers and associated values depending on the omission/inclusion of the  
 1980 'f' query parameter as specified in Section 3.5.1.1.

1981 To retrieve the collection of attribute identifiers for attributes supported by cluster instance <cl> on each  
 1982 endpoint that is a member of group <gid> on the responding device, the requesting device SHALL send a  
 1983 GET request that does not include the 'f' query parameter in the URI.

1984 To retrieve one or more attribute identifiers with their associated values for attributes supported by cluster  
 1985 instance <cl> on each endpoint that is a member of group <gid> on the responding device, the requesting  
 1986 device SHALL send a GET request that includes the 'f' query parameter in the URI.

### 1987 3.16.2 GET Response

1988 A responding device SHALL construct a response with CoAP response code 2.05 Content and with  
 1989 payload containing a map having one entry for each of the device's endpoints that satisfies the following  
 1990 conditions:

- 1991 • The endpoint is a member of group <gid>.
- 1992 • The endpoint supports cluster instance <cl>.

1993 The map entry key SHALL be the endpoint identifier.

1994 If the request URI omits the 'f' query parameter, the map entry value for each included endpoint SHALL  
 1995 be the array of attribute identifiers constructed as specified similarly for a single endpoint in Section 3.5.4.

1996 If the request URI includes the 'f' query parameter, the map entry value for each included endpoint SHALL  
 1997 be the map of attribute entries constructed as specified similarly for a single endpoint in Section 3.5.4.

1998 NOTE: Absence of an entry in an endpoint's attribute map for a particular attribute, which is defined for  
 1999 the cluster and which would otherwise be selected by the 'f' filter, signifies that the attribute is either  
 2000 unsupported by the device, or was inaccessible at the time the request was processed. No further status is  
 2001 provided in the response to differentiate these cases.

### 2002 3.16.3 POST Request

2003 To configure the values of one or more attributes of cluster <cl> on each endpoint that is a member of  
 2004 group <gid> on the responding device such that the configuration of each attribute on an endpoint can  
 2005 succeed or fail independent of the result for the other specified attributes on that endpoint, the requesting  
 2006 device SHALL send a POST request that does not include the 'u' query parameter in the URI.

2007 To configure the values of one or more attributes of cluster <cl> on each endpoint that is a member of  
 2008 group <gid> on the responding device such that none of the attribute values on an endpoint are altered  
 2009 unless all would succeed (undivided write on that endpoint), the requesting device SHALL send a POST  
 2010 request that includes the 'u' query parameter in the URI.

2011 The payload of the request SHALL be a map of key/value pairs. Each key SHALL be an attribute ID and  
 2012 each value SHALL be the value to which the attribute is intended to be set.

### 3.16.4 POST Response

A responding device SHALL process the POST request for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

For each such endpoint, a responding device SHALL process the request as specified similarly for a single endpoint in Section 3.5.6, except that the device SHALL only note for each endpoint the CoAP response code and the payload that is produced, but SHALL NOT construct and send the response messages as specified in that section.

A responding device SHALL construct a response with CoAP response code 2.04 Changed and payload containing a map having one entry for each of the device's endpoints for which processing produced a CoAP response code 2.04 Changed.

The map entry key SHALL be the endpoint identifier.

The map entry value for each endpoint SHALL be the payload constructed as specified for a single endpoint in Section 3.5.6.

For a multicast request, the device SHALL NOT return a response.

### 3.16.5 Group Attribute Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>/<cl>/a	GET		Response Code: 2.05 Content  {1: [0, 65533], 2: [0, 65533]}	
/zcl/g/<gid>/<cl>/a?f=*	GET		Response Code: 2.05 Content  {1: {0: {'v': <value>}, 65533: {'v': <value>}}, 2: {0: {'v': <value>}, 65533: {'v': <value>}}}	
/zcl/g/<gid>/<cl>/a	POST	content {0: 10}	Response Code: 2.04 No Content  -- or -- Response Code: 2.04 Content: {1: {0: {'s': 0x88}}, 2: {0: {'s': 0x88}}}	
/zcl/g/<gid>/<cl>/a	PUT	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/a	DELETE	Any	4.05 (Method Not allowed)	

**Table 34: Group Attribute Collection Access Example**

## 3.17 Group Attribute Instance

The URI Path

/zcl/g/<gid>/<cl>/a/<aid>

provides access to attribute <aid> for cluster instance <cl> on all endpoints that are members of group <gid>.

### 3.17.1 GET Request

To retrieve the value of attribute <aid> of cluster instance <cl> on all endpoints that are members of group <gid> on the responding device, the requesting device SHALL send a GET request.

### 3.17.2 GET Response

A responding device SHALL construct a response with payload containing a map having one entry for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.
- The attribute <aid> of cluster instance <cl> is supported.
- The value of attribute <aid> is successfully retrieved.

The map entry key SHALL be the endpoint identifier and the map entry value SHALL be the payload constructed as specified for a single endpoint in Section 3.6.2.

### 3.17.3 PUT Request

To configure the value of attribute <aid> of cluster instance <cl> on each endpoint that is a member of group <gid> on the responding device, the requesting device SHALL send a PUT request. The request payload SHALL match the payload specified in section 3.6.3.

### 3.17.4 PUT Response

A responding device SHALL process the PUT request for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

For each such endpoint, a responding device SHALL process the request as specified similarly for a single endpoint in Section 3.6.4, except that the device SHALL only note for each endpoint the CoAP response code and the payload that is produced, but SHALL NOT construct and send the response messages as specified in that section.

A responding device SHALL construct a response with CoAP response code 2.04 Changed and payload containing a map having one entry for each of the device's endpoints for which processing produced a CoAP response code 2.04 Changed.

The map entry key SHALL be the endpoint identifier.

The map entry value for each endpoint SHALL be the payload constructed as specified for a single endpoint in Section 3.6.4.

If an endpoint has an empty response it SHALL NOT be included in the response map and if the response map has no entries, for a unicast response, an empty map shall be returned. For a multicast request, the device SHALL NOT return a response.

### 3.17.5 Group Attribute Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>/<cl>/a/<aid>	GET	None	Response Code: 2.05 Content  {1: {<aid>: {'v': <value>}}, 2: {<aid>: {'v': <value>}}}	
/zcl/g/<gid>/<cl>/a/<aid>	POST	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/a/<aid>	PUT	{<aid>: <new value>}	Response Code: 2.04 No Content  --- or --- Response Code: 2.04 content: {1: {<aid>: {'s': 0x88}}, 2: {<aid>: {'s': 0x88}}}	
/zcl/g/<gid>/<cl>/a/<aid>	DELETE	Any	4.05 (Method Not allowed)	

Table 35: Group Attribute Instance Access Example

## 3.18 Group Command Collection

The URI Path

/zcl/g/<gid>/<cl>/c

provides access to the collection of commands for cluster instance <cl> that are available on the endpoints that are members of group <gid>.

### 3.18.1 GET Request

The requesting device SHALL send a GET request to retrieve the collection of commands for cluster instance <cl> on all endpoints that are members of group <gid> of the responding device.

### 3.18.2 GET Response

A responding device SHALL construct a response with payload containing a map having one entry for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

The map entry key SHALL be the endpoint identifier.

The map entry value for each included endpoint SHALL be the array of command identifiers constructed as specified similarly for a single endpoint in Section 3.7.2.

### 3.18.3 Group Command Collection Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>/<cl>/c	GET	None	Response Code: 2.05 Content  content: {1: [0, 1, 2], 2: [0, 1, 2]}	
/zcl/g/<gid>/<cl>/c	POST	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/c	PUT	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/c	DELETE	Any	4.05 (Method Not allowed)	

**Table 36: Group Command Collection Access Example**

## 3.19 Group Command Instance

The URI Path

/zcl/g/<gid>/<cl>/c/<cid>

provides access to a command for cluster instance <cl> on endpoints that are members of group <gid>.

Commands within the group URI-Path zcl/g/<gid> are invoked at all the endpoints members of a group.

Commands that are optional for a cluster SHOULD NOT be executed at the endpoints that do not implement them.

For group command execution via multicast IP addressing, if no endpoint can process the request successfully, no response is returned. If one or more endpoints can process the request successfully, then a response via unicast IP addressing SHALL be returned if there is a response payload.

### 3.19.1 POST Request

The requesting device SHALL send a POST request to invoke command <cid> for cluster instance <cl> on all endpoints that are members of group <gid> of the responding device.

If the ZCL cluster definition specifies a request command payload, the request payload is formed by applying ZCL-to-ZCLIP cluster command payload translation rules, defined in Chapter 8, to the cluster request command payload definition in [ZCL].

### 3.19.2 POST Response

A responding device SHALL process the POST request for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

2112 For each such endpoint, a responding device SHALL process the request as specified similarly for a single  
 2113 endpoint in Section 3.8.3, except that the device SHALL only note for each endpoint the CoAP response  
 2114 code and the payload that is produced, but SHALL NOT construct and send the response messages as  
 2115 specified in that section.

2116 A responding device SHALL construct a response with CoAP response code 2.04 Changed and payload  
 2117 containing a map having one entry for each of the device's endpoints for which processing produced a  
 2118 CoAP response code 2.04 Changed.

2119 The map entry key SHALL be the endpoint identifier.

2120 The map entry value for each endpoint SHALL be the payload constructed as specified for a single  
 2121 endpoint in Section 3.8.3.

2122 If an endpoint has an empty response it SHALL NOT be included in the response map and if the response  
 2123 map has no entries, for a unicast response, an empty map SHALL be returned. Multicast response  
 2124 suppression is defined in 2.8.6.2.

### 2125 3.19.3 Group Command Instance Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>/<cl>/c/<cid>	GET	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/c/<cid>	POST		Response Code: 2.04 Content: {}  --- or --- Response Code 2.04 content: {1: {0: <response field>}, 2: {0: <response field>}}	
/zcl/g/<gid>/<cl>/c/<cid>	PUT	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/c/<cid>	DELETE	Any	4.05 (Method Not allowed)	

2126 **Table 37: Group Command Instance Access Example**

## 2127 3.20 Group Default Report Configuration Instance

2128 The URI Path

2129 /zcl/g/<gid>/<cl>/r/0

2130 provides access to the default report configuration configured for cluster instance <cl> on the endpoints that  
 2131 are members of group <gid>.

2132 No group support for report configuration is provided as the management of non-default reports would be  
 2133 problematic across multiple endpoints and devices due the possibility of different report ids being  
 2134 generated between endpoints and devices.

### 3.20.1 GET Request

The requesting device SHALL send a GET request to retrieve the default attribute reporting configuration for cluster instance <cl> on all endpoints that are members of group <gid> of the responding device.

### 3.20.2 GET Response

A responding device SHALL construct a response with payload containing a map having one entry for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

The map entry key SHALL be the endpoint identifier.

The map entry value for each included endpoint SHALL be the report configuration as specified for a single endpoint in Section 3.12.2.

### 3.20.3 PUT Request

To update the default reporting configuration entry, the device SHALL send a PUT request with the payload set to a map representing the reporting entry. The payload SHALL be constructed as described in Section 3.12.3 except that the Binding Destination URI field SHALL NOT be included.

### 3.20.4 PUT Response

A responding device SHALL construct a response with payload containing a map having one entry for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

The map entry key SHALL be the endpoint identifier.

The device does not return any content in the response. Multicast response suppression is defined in 2.8.6.2.

### 3.20.5 DELETE Request

To reset the default attribute reporting configuration entries for cluster <cid> of the endpoints in group <gid>, the device SHALL send a DELETE request.

### 3.20.6 DELETE Response

A responding device SHALL process the POST request for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

For each endpoint that is processed, the device SHALL remove the existing default report configuration and immediately create a new entry in its place with the factory defaults for the report configuration.



## 3.20.7 Group Default Report Configuration Instance Access Example

URI	Method	Request	Response	Notes
/zcl/g/<gid>/<cl>/r/0	GET		Response Code: 2.05 Content: {<ep1>: {"a": {<a0>: {}}, "n":0, "x": 65534}, <ep2>: {"a": {<a0>: {}}, "n":0, "x": 65534}}	
/zcl/g/<gid>/<cl>/r/0	POST	Any	4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/r/0	PUT	{"a": {0: {"r": <ReportableChange>}}, "n": <minInterval>, "x": <maxInterval>}	Response Code: 2.04	
/zcl/g/<gid>/<cl>/r/0	DELETE		Response Code: 2.02, 4.04 or 4.05	

**Table 38: Group Default Report Configuration Instance Access Example**

## 3.21 Group Notification

The URI Path

/zcl/g/<gid>/<cl>/n

provides access to the group cluster notification resource.

### 3.21.1 POST Request

To notify subscribed group <gid> of an attribute change, the device SHALL send a POST request. The payload SHALL be as specified in Section 3.13.1.

### 3.21.2 POST Response

A responding device SHALL process the POST request for each of the device's endpoints that satisfies the following conditions:

- The endpoint is a member of group <gid>.
- The endpoint supports cluster instance <cl>.

For each such endpoint, a responding device SHALL process the request as specified similarly for a single endpoint in Section 3.13.2, except that the device SHALL only note for each endpoint the CoAP response code but SHALL NOT construct and send the response messages as specified in that section.

A device SHALL construct and transmit a response with CoAP response code 2.04 Changed and no content when the message is received via unicast.

2188 A device SHALL NOT transmit a response when the POST request is received via multicast.

### 2189 3.21.3 Group Notification Access Example

URI	Method	Request Payload	Response	Notes
/zcl/g/<gid>/<cl>/n	GET	Any	Response Code: 4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/n	POST	{ "u": <value>, "r": <value>, "b": <value>, "t": <value>, "a": <map of attribute values> }	Response Code: 2.04 (Changed)  No Content	
/zcl/g/<gid>/<cl>/n	PUT	Any	Response Code: 4.05 (Method Not allowed)	
/zcl/g/<gid>/<cl>/n	DELETE	Any	Response Code: 4.05 (Method Not allowed)	

2190 **Table 39: Group Notification Access Example**

2191

## 4 Discovery

[CORELINK] specifies a service to support resource discovery in a CoAP-based network. ZCLIP uses this service for two purposes:

- UID to IP address resolution for a ZCLIP device.
- Discovery of ZCL resources supported by a ZCLIP device.

This chapter specifies the CoAP-based discovery messaging and the filtering attributes used in ZCLIP address resolution and resource discovery transactions.

The ZCLIP discovery mechanism provides a means for service discovery that is defined by the [CORELINK] document and is meant to cover the following use cases:

- A discovery client can discover all devices in a network that implement ZCLIP.
- A discovery client can discover all the Zigbee endpoints which implement a specific Cluster ID (server/client)
- A discovery client can discover all the Zigbee endpoints which implement a specific version of a Cluster ID (server/client)
- A discovery client can discover all the Zigbee endpoints which match a specific Zigbee Device ID
- A discovery client can discover all the Zigbee endpoints which implement a specific version of a Cluster ID (server/client) and match a specific Zigbee Device ID.
- A discovery client can discover all metadata of all Zigbee resources available on the device.

### 4.1 Discovery Request

#### 4.1.1 Discovery Request URI

The discovery request URI has the form

```
<scheme>://<destination>/well-known/core?attr1=val1&attr2=val2&...
```

where

<code>&lt;scheme&gt;</code>	<i>coap</i> or <i>coaps</i> , depending on whether access is unsecured or DTLS-secured
<code>&lt;destination&gt;</code>	IP unicast or multicast destination
<code>/well-known/core</code>	URI Path of the CoRE discovery entry point
<code>attr1=val1...</code>	URI query string that specifies attribute value filtering to refine the subset of discovery results returned

The response to a request depends on whether the request was directed to a unicast or multicast destination, and whether any attribute value filtering was included in the URI, as specified in subsequent sections.

## 4.1.2 Secure Discovery Access

A device MAY support the unsecured *coap* scheme and SHALL support the DTLS-secured *coaps* scheme for discovery requests directed to it via a unicast destination.

A device SHALL support the unsecured *coap* scheme for discovery requests directed to it via a multicast destination.

## 4.1.3 Attribute Value Filter

A device SHALL support URI query attribute value filtering as specified in [CORELINK], and additionally SHALL support query attribute filters that contain more than one attribute.

NOTE: [CORELINK] specifies support for at most one attribute in a query attribute filter. Support for multiple filter attributes is considered a necessary extension for efficient discovery of ZCLIP resources.

The CoRE link format attributes used in ZCLIP discovery operations are specified in Section 4.3.

The attribute names and values in a URI query attribute filter SHOULD NOT be delimited by quote characters.

A discovery request directed to a unicast destination SHOULD contain a query attribute filter.

A discovery request directed to a multicast destination SHALL contain a query attribute filter.

If a query attribute filter contains more than one attribute, the attributes MAY appear in any order; order has no significance. The effect of a filter that contains more than one attribute SHALL be the Boolean AND of the filtering conditions expressed by each of the individual attributes.

Also note that per [CORELINK] the asterisk '\*' wildcard can be added as the last character of an attribute value:

```
/.well-known/core?attr=<value_prefix>*
```

where <value\_prefix> is an arbitrary value prefix string. The wildcard format will match all attribute values that begin with the specified prefix.

## 4.1.4 Requested Encoding of Discovery Data

A discovery request message made by a device implementing this specification SHALL contain the CoAP Accept option set to the value application/link-format+cbor. If a request is received in which the Accept option is not included, e.g. from a non-zclip device, application/link-format SHALL be assumed.

Devices MAY implement support for application/link-format. Devices that do not support application/link-format SHALL return a response with CoAP status code 4.06 Not Acceptable with no content.

NOTE: IANA has not yet assigned a Content-Format value for application/link-format+cbor. The value is shown in [JSONLINK] as "TBD64", indicating that the value assigned is likely to be 64. Until the assignment is made, the value 65064 shall be used (selected from the Content-Format experimental numbering range 65000-65535).

## 4.1.5 Multicast Addressing for Discovery

The IPv6 address range ff0x::fd is assigned by IANA as the All CoAP Nodes variable scope multicast address [COAP]. The "x" designates the 4 bits that declare the multicast address scope [MC6SCOPE]; all

2257 scopes specified in 2.2.1.3 SHALL be used. The All CoAP Nodes multicast address SHALL be used when  
2258 discovery is performed using multicasting."  
2259

## 2260 4.2 Discovery Response

### 2261 4.2.1 Response Payload Encoding

2262 The CBOR-encoded CoRE Link Format [JSONLINK] content format, identified by the media type  
2263 application/link-format+cbor, SHALL be supported to represent discovery results. Encodings other than  
2264 application/link-format+cbor, as specified in the CoAP Accept option of a discovery request, MAY be  
2265 supported to represent discovery results.

2266 As specified in [JSONLINK], the payload of the response to a successful discovery query SHALL be a  
2267 CBOR array containing zero or more resource links that satisfy the request query filter parameters. Each  
2268 link is represented as a CBOR map of key/value pairs, for which the keys identify the components of the  
2269 link. Per [JSONLINK], certain link components are identified by numeric keys, others by text keys. The  
2270 examples in Annex D.3 illustrate the structure of the payload and the numeric and text keys used.

2271 The exact resource URI and attributes returned depend on the query parameters presented.

2272 The resource URI of each returned link MAY be either an absolute or relative URI.

2273 If and only if the resource URI of a returned link is an absolute URI, the resource URI SHALL have either  
2274 unsecured *coap* or DTLS-secured *coaps* scheme consistent with the security model used in the network.

2275 If the resource URI of a returned link is a relative URI, the resource URI Path SHALL be relative to the  
2276 unicast source of the responding device. Relative URIs SHOULD be returned when applicable to conserve  
2277 message size and bandwidth.

2278 The number of links returned may vary and the number of attributes per link may vary as well.

### 2279 4.2.2 Discovery Message Processing Rules

#### 2280 4.2.2.1 Handling of Unicast Discovery Request Message

2281 A discovery request message received via unicast IP addressing SHALL be handled as described in the  
2282 following sections.

##### 2283 4.2.2.1.1 General Discovery Request Validation

2284 Upon receipt of a request, a device SHALL apply the following validation steps:

- 2285 1. The device SHALL check for the existence of the requested resource. If the requested resource  
2286 does not exist on the device a 4.04 Not Found SHALL be returned and processing SHALL cease.
- 2287 2. The device SHALL check that the method requested is GET. If the method is not GET, the device  
2288 SHALL respond with a 4.05 Method Not Allowed response and processing SHALL cease.
- 2289 3. If the request contains a CoAP Accept option, the device SHALL check that the device can  
2290 generate a response using the format specified by the option. If the device is unable to generate a  
2291 response in the format the remote device has specified it can accept, it SHALL return a 4.06 Not  
2292 Acceptable and processing SHALL cease.
- 2293 4. If the request contains a payload, the device SHALL return a 4.00 Bad Request and processing  
2294 SHALL cease.

#### 4.2.2.1.2 Discovery Request-Specific Validation and Execution

Upon successful general request validation, the device SHALL process the request as specified for the URI and apply the following execution validation:

- If an error is detected with the contents of the request (e.g., invalid URI query parameter syntax) the device SHALL return a 4.00 Bad Request message.
- If the content of the request is valid, but the server cannot process it at this time (e.g., temporary overload), the device SHALL return a 5.03 Service Unavailable message.
- If the content of the request is valid, but the server fails to process it because of an unexpected condition, the device SHALL return a 5.00 Internal Server Error message.
- Otherwise, the device SHALL execute the request-specific method and return a response indicating success. Note that if the request is sent to /.well-known/core with no query filter, the request succeeds and the device SHALL return the link attributes corresponding to the /zcl resource URI.

#### 4.2.2.1.3 Response Generation for Successful Discovery Request

When generating a URI-specific response for a successfully executed discovery request, the device SHALL apply the following rules:

1. If the result set contains no links, the payload SHALL be an empty list.
2. If the response includes a payload the device SHALL encode the payload using the format specified by the CoAP Accept option, if that option is provided by the client, otherwise using 'application/link-format+cbor', and place this into the CoAP payload. The device SHALL add a CoAP Content-Format option set to the encoding type used.
3. If the result set exceeds the size of a single CoAP response datagram, the device SHALL set the CoAP Block2 option.
4. The device SHALL set the code of the CoAP response to 2.05 Content.

The device SHALL return the discovery response.

#### 4.2.2.2 Handling of Multicast Discovery Request Message

A request message received via multicast IP addressing SHALL be processed as described for a request message received via unicast IP addressing in Section 4.2.2.1, but with the following differences.

Request processing SHALL cease and a response message SHALL NOT be sent if any of the following conditions are detected:

1. The CoAP server supports a mechanism to authenticate and secure the multicast request, and the request fails to be authenticated and secured.
2. The discovery request URI did not include a query attribute value filter.
3. The code for the response is 4.xx or 5.xx indicating processing of the request failed.
4. The code for the response is 2.xx indicating that processing of the request succeeded; and the response has an empty payload (i.e. no results found for the discovery request).

If a response is to be sent, the device SHALL delay sending its response as described in Section 2.2.3.3.

Response messages SHALL be sent using unicast IP addressing.

### 4.2.2.3 Handling of Discovery Response Message

Upon receipt of a response to a discovery request a device SHOULD perform the following steps:

The device SHOULD check the CoAP code to determine if the request was executed successfully on the remote device.

- A 2.05 code indicates that the device was able to successfully process the request and that the payload contains the discovery response (which may be empty).
- A 4.xx code indicates that there was a problem with the request and that the device must modify it before it can be successfully processed.
- A 5.xx code indicates that while the request was valid, the server could not process it at the time (e.g., operating constraint). The device MAY attempt to retry the request at a later time.

The device SHOULD check the Content-Format option to ensure that the payload is encoded in a format that it is capable of processing before attempting to process and use the information contained in the response.

The device SHALL check if the CoAP Block2 option is present, and if so, SHALL handle the response as described in Section 2.8.6.

## 4.3 Resource Attributes Used in ZCLIP

This section specifies the URN resource naming and resource attributes used by ZCLIP discovery operations. Illustrative examples of discovery queries using these attributes are found in Annex D.3. The valid combinations of attribute values that may be used to query a resource are shown in Table 40. Each of the link attributes shown in the table are described in later sections.

Resource URI	Link Attribute			
	rt	if	ep	ze
/zcl	X	X (zcl version)	X	
/zcl/e/<eid> (zcl endpoint)		X (device type version)		X
/zcl/e/<eid>/<cl> (zcl cluster)	X	X (cluster version)		X

**Table 40. Valid Resource URI / Link Attribute Combinations**

### 4.3.1 Attribute Value URN Namespace

As described in [URN] Section 2, all URNs have the syntax

<URN> ::= "urn:" <NID> ":" <NSS>

where <NID> is the Namespace Identifier and <NSS> the Namespace Specific String.

2359 The Zigbee Alliance will register the Namespace Identifier urn:zcl in the IANA Registry of URN  
 2360 Namespaces and will hold administrative authority of the URN namespace under that identifier. Resource  
 2361 attribute values used for ZCLIP discovery operations are allocated from that namespace.

## 2362 **4.3.2 Representation of Numeric Values**

2363 Numeric values contained in a resource attribute value SHALL be expressed in Base 16 (hexadecimal).  
 2364 Hexadecimal digits corresponding to Base 10 values 10, 11, 12, 13, 14, 15 SHALL be represented by  
 2365 lowercase characters 'a', 'b', 'c', 'd', 'e', 'f' respectively.  
 2366 Numeric values consisting of two or more hexadecimal digits SHALL NOT contain leading zeroes.  
 2367

## 2368 **4.3.3 CoRE Standard Resource Attributes**

### 2369 **4.3.3.1 Resource Type 'rt'**

2370 The Resource Type attribute 'rt' is defined in [CORELINK] Section 3.1.  
 2371 For ZCLIP, the 'rt' attribute value is used to filter a discovery request based on entry-point resource or ZCL  
 2372 cluster resource. The syntax of the value of this attribute follows the rule:

rt=urn:zcl	for entry-point resource (i.e. /zcl)
rt=urn:zcl:c.<cluster ID>.<side>	for ZCL cluster resource (i.e. /zcl/e/<eid>/<cl>)

2373 where urn:zcl is the namespace root, 'c' signifies a cluster resource, <cluster ID> is the numeric cluster  
 2374 number as specified in [ZCL], and <side> is either 'c' or 's' representing client and server cluster side,  
 2375 respectively.

2376 When represented in a resource link CBOR map, the key for this link attribute is numeric 9 decimal.

### 2377 **4.3.3.2 Interface Description 'if'**

2378 The Interface Description attribute 'if' is defined in [CORELINK] Section 3.2.  
 2379 For ZCLIP, the 'if' attribute MAY be used in conjunction with a resource type attribute or a Zigbee  
 2380 endpoint attribute to specify a version number for the device type respectively. The syntax of the value of  
 2381 this attribute is:

if=urn:zcl:v<#>	for entry-point resource (i.e. /zcl)
if=urn:zcl:d.v<#>	for ZCL endpoint resource (i.e. /zcl/e/<eid>)
if=urn:zcl:c.v<#>	for ZCL cluster resource (i.e. /zcl/e/<eid>/<cl>)

2382 where urn:zcl is the namespace root, 'v' signifies version, and <#> is the numeric version number, , 'd:v'  
 2383 signifies device type (such as on/off light, dimmable light, occupancy sensor, etc.) version and 'c:v'  
 2384 signifies a cluster resource version.

2385 When represented in a resource link CBOR map, the key for this link attribute is numeric 10 decimal.

### 2386 **4.3.3.3 Endpoint Name 'ep'**

2387 The Endpoint Name attribute 'ep' is defined in [CORERD] Section 5.3.



NOTE: The term "endpoint" here comes from [CORERD] which in turn refers to [COAP]'s definition of endpoint as an entity participating at the CoAP protocol layer; not a (higher layer) endpoint as defined by Zigbee.

Endpoint Name provides, in the context of RD, an identifier of a CoAP endpoint that is unique within a defined Domain. The value of the Endpoint Name attribute might be a UID, a device serial number or a human-readable unique name. A Domain might be a single site of installation, or the entire world (global).

For ZCLIP, the 'ep' attribute value SHALL be the UID specified in Section 2.6.4. The discovery resource attribute value of a UID is the Named Information URI format defined for the SHA-256 hash in [HASHNAME]:

```
ni:///sha-256;<hash value>
```

where the <hash value> is the 256-bit UID value in network byte order, encoded as a 43 character base64url encoded string, with no '=' characters.

When returning the UID in the Named Information URI format, it SHALL be enclosed in quotes for application/link-format. Other uses such as application/link-format+cbor or query string usage SHALL NOT be enclosed in quotes.

When represented in a resource link CBOR map, the key for this link attribute is text "ep".

## 4.3.4 Zigbee Alliance-Defined Resource Attributes

### 4.3.4.1 Zigbee Endpoint 'ze'

To distinguish between Zigbee endpoints implementing the same cluster, the Zigbee Endpoint attribute 'ze' is hereby defined. The presence of this attribute in a discovery response provides an explicit representation of the Zigbee endpoint. If not present, the Zigbee endpoint can be derived from the <eid> element in the resource URI(s) returned in the discovery response. The syntax of the value of the 'ze' attribute is:

```
ze=urn:zcl:d.<device ID>.<endpoint ID>
```

where urn:zcl is the namespace root, 'd' signifies device type, <device ID> is the numeric device type identifier (such as on/off light, dimmable light, occupancy sensor, etc.), and <endpoint ID> is the numeric endpoint identifier of a Zigbee endpoint that implements that device type.

The 'ze' attribute SHOULD always be present in the CoRE link format response for each entry representing a resource at the Zigbee endpoint level or below (e.g. endpoints, clusters). Responses to queries to discover higher-level ZCLIP entry points (i.e. /zcl) SHALL NOT include the 'ze' attribute.

When represented in a resource link CBOR map, the key for this link attribute is text "ze".

## 4.4 Resource Directory (RD)

A CoRE Resource Directory (RD) as defined in [CORERD] MAY be present in the network. The RD represents a central repository which provides directory services where devices can register their services/properties and other devices can discover these through query functions. An RD facilitates discovery functions for nodes on constrained, bandwidth-limited mesh networks and for discovery of sleepy nodes.

In case of presence of a central RD in the network, devices SHALL register at the RD and SHALL NOT respond to discovery queries.

## 4.4.1 RD Interfaces

The RD provides separate interfaces for the registration of resources and the lookup (i.e. discovery) of resources. The RD registration and lookup interfaces SHALL be accessed via unicast communication. Any request to the RD registration and lookup interfaces via multicast communication SHALL be ignored.

### 4.4.1.1 Registration Interface

The RD registration interface is used by ZCLIP devices to register discovery information available in the system at the RD. Devices are required to register information about themselves.

The registration interface SHALL support the content format "application/link-format+cbor".

This interface SHALL be secured in accordance with the specifications in Section 4.4.6.

### 4.4.1.2 Lookup Interface

The RD lookup interface is used by devices to retrieve information about resources.

Although IETF RD [CORERD] lookup function set (interface) provides additional capabilities to discover endpoints, domains or groups, only the resource lookup interface is adopted in this specification to keep the compatibility with the queries to the distributed discovery via the "/.well-known/core" interface.

The lookup interface SHALL support the content format "application/link-format+cbor".

The lookup interface SHALL be located at "/rd-lookup" on the identified interface.

This interface SHALL be secured in accordance with the specifications in Section 4.4.6.

## 4.4.2 Discovery and Configuration of RD interfaces

Devices SHALL support discovery of and registration with a RD as described in the next sections. The RD SHALL advertise its interfaces via unicast and multicast discovery.

### 4.4.2.1.1 Resource Directory Configuration Entry

A "RD\_entry" element consists of a map which SHALL contain the entries specified in <Table Ref>.

Name	Key	Type	Encoding	Default
Resource Directory URI	"rd"	Character string	Major type 3	N/A
Registration Status Indicator	"r"	Integer	Major type 0	0

#### 4.4.2.1.1.1 Registration Directory URI

The registration directory URI is provided with the key "rd" and a string value containing the URI of a RD according in the following format:

*<URI-Scheme>://<URI-Host>:<URI-Port>*

where:

- 2454 • *<URI-Scheme>* represents the application layer protocol used (i.e. *coaps*).
- 2455 • *<URI-Host>* represents the IPv6 address or hostname of the RD.
- 2456 • *<URI-Port>* represents the transport-layer port number of the RD. (optional)

2457 The URI path is not included in an URI in the RD\_entry and it SHOULD be ignored if provided.

#### 2458 4.4.2.1.1.2 Registration Status Indicator

2459 The registration status indicate is provided with the key "r" and an unsigned integer value containing a  
 2460 status indicator indicating the current registration status of the device with respect to the Registration  
 2461 Directory URI. The allowable values are specified in <Table Ref>. All references to resource directory in  
 2462 the table refer exclusively to the resource directory which is specified by the Resource Directory URI in the  
 2463 same RD\_entry.

Value	Name	Description
0	Not Registered	The device does not have any registration with the resource directory. Deletion of the entry is possible while in this state.
1	Registered	The device is currently registered with the resource directory. Deletion of the entry is not possible while in this state.
2	Performing Registration	The device is currently in the process of registering with the resource directory. Deletion of the entry is not possible while in this state.
3	Removing Registration	The device is currently in the process of removing its registration from the resource directory. Deletion of the entry is not possible while in this state.

2464

#### 2465 4.4.2.1.2 Resource Directory Configuration Collection Resource

2466 The URI Path /rd/conf provides access to the collection of resource directory configurations on the device.

##### 2467 4.4.2.1.2.1 GET Request

2468 The requesting device SHALL send a GET request to retrieve the collection of RD\_entry keys configured  
 2469 on the responding device.

##### 2470 4.4.2.1.2.2 GET Response

2471 The responding device SHALL return the map containing the serialized key values and corresponding  
 2472 RD\_entry elements.

#### 4.4.2.1.2.3 POST Request

To create a new RD\_entry element a requesting device SHALL send a request via POST method to the /rd/conf resource containing the requested RD\_entry element in the payload.

#### 4.4.2.1.2.4 POST Response

The responding device SHALL process the incoming message in accordance with the general rules specified in Section 2.8.6.

The device SHALL validate that the registration status indicator is either omitted or set to a value of Not Registered (0). In the case that the indicator is set to any other value, the device SHALL respond with a 4.00 Bad Request.

If no errors occur, the responding device SHALL respond with Location-Path option set to /rd/conf/<rdid>, where <rdid> is the id of the new RD\_entry. The device SHALL then proceed to start registration with the RD specified in the RD\_Entry.

#### 4.4.2.1.2.5 Resource Directory Configuration Collection Access Example

URI	Method	Request Payload	Response	Notes
/rd/conf	GET	Any	Response Code: 2.05 Content Content: {1: {"rd": "coaps://[aaaa::1234:2345]:5683"}}	
/rd/conf	POST	{"rd": "coaps://[aaaa::1234: 2345]:5683"}	Response Code: 2.01 Created Location-Path: /rd/conf/0	
/rd/conf	PUT	Any	4.05 (Method Not allowed)	
/rd/conf	DELETE	Any	4.05 (Method Not allowed)	

**Table 41: Resource Directory Configuration Collection Access Example**

#### 4.4.2.1.3 Resource Directory Configuration Instance Resource

The URI Path /rd/conf/<rdid> provides access to a resource directory configuration instance on the device.

The Registration Status Indicator is managed by the device, and reflects the current registration status for a specific instance. Upon creation of a resource directory configuration instance, the device SHALL start the process of registration with the resource directory specified in the Resource Directory URI parameter of the instance.

To instruct a device to deregister, a PUT request SHALL be sent to the appropriate /rd/conf/<rdid> entry with the Resource Directory URI unchanged and the Registration Status Indicator set to Removing Registration (3).

To delete an RD\_entry element a requesting device SHALL send a request via DELETE method to the /rd/conf/<rdid> resource containing the RD\_entry element identified by <rdid> ID to be deleted. The responding device SHALL process the incoming message in accordance with the general rules specified in Section 2.8.6. The device SHALL validate that the Registration Status Indicator is set to Not Registered. In the case that the indicator is set to any other value, the device SHALL respond with a 4.00 Bad Request. If no errors occur, the responding device SHALL permanently remove the RD\_entry.

#### 2502 4.4.2.1.3.1 GET Request

2503 The requesting device SHALL send a GET request to retrieve the RD\_entry details for the resource on the  
2504 responding device.

#### 2505 4.4.2.1.3.2 GET Response

2506 The responding device SHALL return the map representing the RD\_entry with id <rdid>.

#### 2507 4.4.2.1.3.3 PUT Request

2508 To create a new RD\_entry element a requesting device SHALL send a request via PUT method to the  
2509 /rd/conf/<rdid> resource containing the updated RD\_entry element in the payload.

#### 2510 4.4.2.1.3.4 PUT Response

2511 The responding device SHALL process the incoming message in accordance with the general rules specified  
2512 in Section 2.8.6.

2513 The device SHALL validate that the Resource Directory URI in the request matches the Resource Directory  
2514 URI stored in the resource when the Registration Status Indicator stored in the has any value other from Not  
2515 Registered. If the validation fails, the device SHALL respond with a 4.00 Bad Request.

2516 The device SHALL validate that the Registration Status Indicator is permitted to transition from the stored  
2517 registration state value to the value provided in the request. The valid transitions via a PUT request are marked  
2518 with an X in Table 43. The device SHALL not permit other transitions to occur via a PUT request. If a  
2519 disallowed transition is attempted, the device SHALL respond with a 4.00 Bad Request.

Target Present	Not Registered	Registered	Performing Registration	Removing Registration
Not Registered	X		X	X
Registered		X	X	X
Performing Registration			X	X
Removing Registration				X

2520 **Table 42: Allowable Registration Status Transitions**

2521 If no errors occur, the responding device SHALL respond with a status code of 2.04 Changed and No Content.  
2522 The device SHALL then proceed to transition to the registration state in the request if not already in that  
2523 state.

#### 2524 4.4.2.1.3.5 DELETE Request

2525 The requesting device SHALL send a DELETE request to remove the RD\_entry on the responding device.

#### 2526 4.4.2.1.3.6 DELETE Response

2527 The responding device SHALL validate that the Registration Status Indicator of the RD\_entry identified by  
2528 <rdid> is Not Registered. The device SHALL not to occur in other states. If a disallowed transition is  
2529 attempted, the device SHALL respond with a 4.00 Bad Request. If the resource is successfully deleted, the  
2530 device SHALL respond with a 2.02 Deleted. In both cases, there is no payload.

#### 4.4.2.1.3.7 Resource Directory Configuration Collection Access Example

URI	Method	Request Payload	Response	Notes
/rd/conf	GET	Any	Response Code: 2.05 Content Content: {"rd": "coaps://[aaaa::1234:2345]:5683"}	
/rd/conf	POST	Any	4.05 (Method Not allowed)	
/rd/conf	PUT	Any	2.04 Changed / 4.00 Bad Request	
/rd/conf	DELETE	Any	2.02 Deleted / 4.00 Bad Request	

**Table 43: Resource Directory Configuration Collection Access Example**

### 4.4.3 Registration of Resources at the RD

The registration of resources at RD leverages the registration interface defined in Section 4.4.1.1 and it is carried out based on the method defined in [CORERD], Section 5.3.

A device registers its resources using the registration interface. Devices SHALL only register their own resources.

A device that has a resource successfully registered with the RD SHALL NOT respond to discovery queries.

After a failed attempt to update a registration resource, a device SHALL re-enable the responses to incoming discovery requests on the ./well-known/core interface.

Support for "Simple Registration" as defined in [CORERD], Section 5.3.1 SHALL be disabled in this specification to avoid security and scalability risks due in example to extensive discovery requests in a constrained network.

#### 4.4.3.1 Registration Procedure

After discovering the location of an RD, a device MAY register its resources by sending a request via POST method according to the following URI format:

```
<scheme>://<destination>/rd?ep=<ep_name>&lt=<lifetime>&con=<con_val>
```

where:

<scheme>                    *coaps*, DTLS-secured (mandatory)

<destination>            IPv6 address or hostname of the RD (mandatory)

<ep\_name>                 Endpoint name as specified in Section 4.3.3.3 (mandatory).

<lifetime>                 Lifetime of the registration in seconds (optional). Range of 60-4294967295. If no lifetime is included in the initial registration, a default value of 86400 (24 hours) SHALL be assumed. If no update is done within this period, the registration is removed from the RD. If the lt parameter is not included in a

registration refresh or update operation, the most recently supplied value SHALL be re-used.

<context>

Context (optional). This parameter sets the scheme, address and port at which this server is available in the form  
 <server\_scheme>://<server\_IP>:<server\_port>/<server\_path>. If omitted, the scheme of the protocol, source IPv6 address and source port of the register request are assumed. If the parameter is set, <server\_scheme> and <server\_host> parameters SHALL be specified. <server\_port> and <server\_path> parameters MAY be specified. If a device uses an ephemeral port to register with, it SHALL include the con parameter in the registration to provide a valid network path.

2552

2553 The Content-Format of the request SHALL be set to application/link-format+cbor.

2554 The payload of the resource registration request SHALL be a CBOR array containing zero or more  
 2555 resource links. Each link is represented as a CBOR map of key/value pairs, for which the keys identify the  
 2556 components of the link. Per [JSONLINK], certain link components are identified by numeric keys, others  
 2557 by text keys.

2558 The payload of the resource registration request might be larger than the size of the payload of a single  
 2559 CoAP datagram. In these cases, a CoAP Block1 option SHALL be used to signal the desire for a block  
 2560 based transfer as described in [COAPBLK]. The RD SHALL support Block1 transfer.

2561 The responding device SHALL process the incoming message in accordance with the general rules  
 2562 specified in Section 2.8.6. If the service could not perform the operation, it SHALL respond with 5.03  
 2563 Service Unavailable. If no errors/failures occur, the responding device SHALL respond including the  
 2564 Location-Path option set to /rd/<regID>, which represents the new registration entry. This Location  
 2565 SHALL be a stable identifier generated by the RD as it is used for all subsequent operations on this  
 2566 registration.

2567

2568 Examples of interactions to register resources are shown below.

2569 Req: POST /rd?ep=ni:///sha-256;<hash value>

2570 Content-Format: application/link-format+cbor

2571 Payload:

2572 [{1: "/zcl", 9: "urn:zcl", 10: "urn:zcl"}, {1: "/zcl/e/1", 9:  
 2573 "urn:zcl:d", 10: "urn:zcl:d.v1", "ze": "urn:zcl:d.101.1"}, {1:  
 2574 "/zcl/e/1/s6", 9: "urn:zcl:c.6.s", 10: "urn:zcl:c.v1", "ze":  
 2575 "urn:zcl:d.101.1"}, {1: "/zcl/e/1/c6", 9: "urn:zcl:c.6.c", 10:  
 2576 "urn:zcl:c.v1", "ze": "urn:zcl:d.101.1"}]

2577 Res: 2.01 Created

2578 Location: /rd/4521

#### 2579 4.4.3.2 Registration Update

2580 A device MAY send a request to update a registration resource via a POST on the registration resource,  
 2581 following the definition in [CORERD], Section 5.4.1.



2582 The update will in most cases just renew the lifetime, keeping the registration valid. It also allows to change  
 2583 existing registrations, e.g. to add attributes.

2584 A device **SHOULD** update a registration resource before the expiration of its lifetime.

2585 Devices **SHALL** update the registration resources following a change to their IPV6 address.

2586 Req: POST /rd/4521?lt=43200

2587 Res: 2.04 Changed

#### 2588 4.4.3.3 Registration Removal

2589 A device **SHOULD** remove its entry from the RD if it knows it will no longer be available (for example on  
 2590 shutdown). This is accomplished by performing a DELETE on the registration resource following the  
 2591 definition in [CORERD], Section 5.4.2.

2592 Req: DELETE /rd/4521

2593 Res: 2.02 Deleted

#### 2594 4.4.4 RD Lookup

2595 Discovery using RD is carried out by performing queries to the RD lookup interface, identically to the  
 2596 queries supported on the "/.well-known/core" interface, i.e. according to the definitions in Section 4.1. For  
 2597 this purpose, this specification makes the lookup of resource according to the Resource Lookup (as defined  
 2598 in [CORERD] 7.1) mandatory part of the RD implementation. Responses to queries to the RD look up  
 2599 interface follow the definitions in Section 4.2. The resource attributes used by RD discovery follow the  
 2600 definitions in Section 4.3, with the exception that 'ep' link attribute corresponding to the UID of the device  
 2601 a resource belongs to **SHALL** be included in the response to a discovery request for each resource  
 2602 registered at the RD.

2603

2604 An example of RD lookup interaction is shown below:

2605 Req: GET /rd-lookup/res?rt=urn:zcl:c.3.s

2606 Res: 2.05 content

2607 [{1: "/zcl/e/2/s3", 9: "urn:zcl:c.3.s", 10: "urn:zcl:c.v1", "ze": "urn:zcl:d.106.2", "ep": "ni:///sha-  
 2608 256;NvbKUrHKwCpPamtO80GvjPloHyUBWwgfhOIEpS1bU"}, {1: "/zcl/e/1/s3", 9: "urn:zcl:c.3.s", 10:  
 2609 "urn:zcl:c.v1", "ze": "urn:zcl:d.102.1", , "ep": "ni:///sha-  
 2610 256;NvbKUrHKwCpPamtO80GvjPloHyUBWwgfhOIEpS1bU"}]

2611

#### 2612 4.4.5 Support for Multiple Resource Directories

2613 Multiple RDs **MAY** be implemented to avoid single points of failure and to allow load balancing. In a  
 2614 system with multiple RDs implemented, multiple RDs need to provide consistent information on their  
 2615 lookup interfaces. All registration interfaces available from multiple RDs need to provide identical  
 2616 functionality and therefore a device needs to register only to one of them.

2617

2618 Any change to the content of any of the respective RDs needs to be synchronized with all other RDs. This  
 2619 specification does not detail the synchronization method used by RDs, but the synchronization method



2620 needs to guarantee that the links returned after registration are valid and non-conflicting for the whole  
2621 multi-RD system.

## 2622 **4.4.6 Security**

2623 Malfunctioning or compromised RD have effects that affect potentially the full installation in the building.  
2624 The main risks seen are:

- 2625     • Rogue entity registers, updates or deletes information pertaining to other nodes
- 2626     • Rogue entity impersonates as the RD and either responds to RD lookup or registration queries by  
2627       providing falsified information or losing information
- 2628     • Rogue entity modifies queries to the RD lookup interface thus falsifying the information provided  
2629       back (man in the middle)
- 2630     • Rogue entity reads out the RD, thus acquiring info on the availability of nodes and resources in the  
2631       network

### 2632 **4.4.6.1 Solutions**

2633 Devices SHALL only communicate via DTLS with an RD which has been authenticated according to the  
2634 methodology defined in Chapter 6.

2635  
2636 Devices SHALL use DTLS to communicate with the registration and lookup interfaces of an RD. Devices  
2637 registering, updating or removing resources SHALL be authenticated at the RD based on their UID and  
2638 DTLS session using RawPublicKey or operational certificate.

2639  
2640 Application security access control SHALL be implemented for look up of protected cluster resources (as  
2641 defined in Section 6.5) and specific protected resources using access tokens according to the methodology  
2642 defined in Section 6.6.

## 2643 **4.5 Sleepy Devices**

2644 If an RD is present in the network a sleepy device SHALL send its discovery request to the RD.

2645 Sleepy devices SHALL register with Resource Directory (if present).

2646 Sleepy devices do not reliably respond to discovery requests depending on their sleep cycle. As such the  
2647 normal behavior of sleeping devices is that they discover resources or devices of interest when they join the  
2648 network. However, use of a resource directory provides a means to discover sleepy devices. Sleepy  
2649 devices SHALL register with an RD if it is present on a network. If a sleepy device updates any  
2650 information it SHALL update its registration with the RD.

2651 The sleepy device SHALL register with a lifetime appropriate to its sleep cycle and update this lifetime on  
2652 the RD prior to its expiration.

2653 (Note: In the current Zigbee specification there is text to support discovery of a sleepy device by a parent  
2654 device; but this feature is unused).

2655

2656

## 5 EZ-Mode Commissioning

2657

EZ-Mode Commissioning is outside the scope of the 1.0 release of this specification.

## 6 Security

Security is an important item for connected devices. The use of the Zigbee Cluster Library over IP networks could result in new security concerns because of the potential for increased connectivity to external networks. Existing Zigbee devices have limited requirements for application level security above the Zigbee network security but increased application level security is considered a requirement for IP connected devices.

The following are the mandatory minimum security requirements for devices that are detailed in this section:

- All devices SHALL have a either a device certificate from the manufacturer or self-generate a public/private key pair to provide identity and to be used to derive a unique ID.
- All devices SHALL support handling of operational certificates if the network supports provisioning and usage of such certificates.
- All devices SHALL have a certificate or public key from their manufacturer to validate code signing for upgrade images from the manufacturer.
- All devices SHALL support OTA software update and the update must be with signed images to guarantee integrity and authenticity of the update.
- All devices SHALL support application level DTLS sessions for either device to device or device to cloud interactions. The DTLS session can be based on preshared key, raw public key or certificates.
- A device SHALL be able to restrict access to particular attributes and/or commands based on application level security.
- When network level security is available, it SHALL be used.

The sections below detail the specific requirements for implementation.

### 6.1 Security Suites

[COAP] requires devices to support NoSec and RawPublicKey modes of security with the use of CoAP (section 9: Securing CoAP). For use of the Zigbee Clusters and CoAP the following is required. Peer to peer or end to end application security using CoAP are based on DTLS and using one of the following known mechanisms:

1. Certificate based – MANDATORY to support using X.509 certificates
2. Raw public key between devices – MANDATORY to support
3. Preshared secret between devices – MANDATORY to support

The specific security suites on the server side are as follows and in priority order:

1. TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 using the NIST P-256 prime curve (also known as secp256r1 curve) and X.509 certificate (device or operational).
2. Same as (1) but using raw public key extensions as detailed in [RPKDTLS].
3. TLS\_PSK\_WITH\_AES\_128\_CCM\_8 as defined in [AESTLS]

Note: devices can support additional security suites but they are optional.

Note that the use of raw public keys is susceptible to man in the middle attacks as noted in [RPKDTLS]. This is also true of PSK if the method to insert the PSK is not secure. For the use of anything other than device certificates, handling of an operational certificate on a device, exchange of public keys or PSK

2698 SHALL only be done using a secure session such as DTLS with an authorized device such as a  
2699 commissioning device or trust anchor.

2700 For example, if a device is authenticated onto a network using DTLS, it can be provided an operational  
2701 certificate for that network in that DTLS session and it can trust the authorizations provided in that  
2702 operational certificate. If that DTLS session provided a public key or pre-shared key from another device  
2703 in the network that is also suitable for use to provide application level security. If a device is provided  
2704 these credentials in an unsecure session it should reject the credentials.

2705

## 2706 6.2 DTLS Security

2707 All devices shall support DTLS security as defined in [DTLS].

### 2708 6.2.1 Security Mode

2709 As mentioned in section 6.1 the device SHALL support the following security modes as specified in  
2710 [COAP]:

- 2711 1. Certificate
- 2712 2. RawPublicKey
- 2713 3. PreSharedKey

#### 2714 6.2.1.1 Certificate Format

2715 The certificate format is used as defined in [X509CERT].

2716 The Certificate security mode uses an X.509 certificate that SHALL contain the mandatory fields as  
2717 specified in [X509CERT] section 4.1, including:

- 2718 • tbsCertificate ; which includes:
  - 2719 ○ subjectPublicKeyInfo
  - 2720 ○ issuer
  - 2721 ○ subject ; the subjectName
  - 2722 ○ validity
  - 2723 ○ version ; this SHALL be version 3
  - 2724 ○ serialNumber
  - 2725 ○ signature ; this SHALL be set to AlgorithmIdentifier ecdsa-with-SHA256 [X509PKI]
- 2726 • signatureAlgorithm ; this SHALL be set to AlgorithmIdentifier ecdsa-with-SHA256 [X509PKI]
- 2727 • signatureValue

2728 CommonName or subjectAltName can be present but SHALL NOT be used to verify the identity of the  
2729 device. For verification of device identity refer to section 2.7.

2730 Extensions as defined in [X509CERT] MAY be present in addition to the mandatory fields.

2731 The certificate hash algorithm SHALL be based on sha-256.

### 2732 6.2.2 Cipher Suites Selection

2733 The device SHALL support the following cipher suites:

- 2734 • TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 (Certificate and RawPublicKey mode)
- 2735 [ECCTLS]
- 2736 • TLS\_PSK\_WITH\_AES\_128\_CCM\_8 (PreSharedKey mode) [AESTLS]

2737 Additional cipher suites MAY be supported by the device.

2738 Table 44 shows some possible pairings of DTLS Security Mode and Cipher Suite in descending order of  
 2739 security level, including some with optional cipher suites. When determining the security level to use, a  
 2740 device SHALL apply the following rules in the order shown:

- 2741 1. A certificate SHALL be preferred over a raw public key.
- 2742 2. A raw public key SHALL be preferred over a pre-shared key.
- 2743 3. The encryption algorithm SHALL use a key strength of at least 128 bits.
- 2744 4. The encryption algorithm SHALL be selected based on the highest bit strength shared between
- 2745 both devices.

2746

Security Level	DTLS Security Mode	Cipher Suite
1 (Highest)	Certificate	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
2	Certificate	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
3	RawPublicKey	TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
4	RawPublicKey	TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
5	PreSharedKey	TLS_PSK_DHE_WITH_AES_256_CCM_8
6	PreSharedKey	TLS_PSK_DHE_WITH_AES_128_CCM_8
7	PreSharedKey	TLS_PSK_WITH_AES_256_CCM_8
8 (Lowest)	PreSharedKey	TLS_PSK_WITH_AES_128_CCM_8

2747 **Table 44. Pairings of DTLS Security Mode and Cipher Suite**

## 2748 6.2.3 Curve Selection

2749 As mentioned in Section 6.1, the device SHALL support the "secp256r1" (NIST P-256) curve [NIST]. The  
 2750 device SHALL support the "Supported Elliptic Curves Extension" [AESTLS-2]. The device MAY support  
 2751 other curves. If the device supports other curves, it SHALL support a RawPublicKey or Certificate based  
 2752 on these curves.

## 2753 6.2.4 Point Format Selection

2754 The device SHALL support the "uncompressed point format" [AESTLS-2]. The device SHALL support the  
 2755 "Supported Point Formats Extension" [AESTLS-2]. The device MAY support other point formats. If the  
 2756 device supports other point formats, it SHALL support a RawPublicKey or Certificate based on these point  
 2757 formats.

## 6.2.5 Signature Format Selection

The device SHALL support the "sha-256" hash and "ecdsa" signature algorithm [TLS]. The device SHOULD support the "Signature Algorithm Extension" [TLS].

## 6.2.6 Operational Certificate Support

If the network supports provisioning and usage of an operational certificate, the device SHALL use the provisioned operational certificate for DTLS based communication. When provisioned with an operational certificate, the device SHALL disallow the use of PreSharedKey and RawPublicKey security for unicast communication.

When provisioned with an operational certificate, the device SHALL use the subjectPublicKeyInfo of the operational certificate as basis for the UID (section 2.7).

## 6.2.7 Operational Certificate Provisioning

The method of provisioning security material is not covered in this specification. For ZCLIP 1.0, Operational certificates SHALL be provisioned by an out of band mechanism.

## 6.2.8 DTLS Session Support

Each device SHALL support at least 2 cached DTLS sessions one as a client and one as a server.

## 6.3 Software Updates

All devices SHALL support over the air software updates for their application using the standard OTA update function. All update images SHALL be signed by the manufacturer. The device SHALL validate the image signing for integrity and authenticity of the updated image using the signing certificate that was installed at manufacturing prior to using the upgrade image. The device may do other validation of the image to ensure it is for the proper device and hardware version for example prior to using the upgrade image.

This requirement for use of OTA update functionality applies to the portion of the device covered by this standard and applies no matter what transport is used to provide the upgrade image to the device. To clarify using examples:

- A stand-alone device in a network may receive an upgrade image from an on-network local server or from a cloud service. This update may include the application and the communication stack but could only be the application image. The device must properly validate the image prior to use and execute the update when commanded.
- A device with another network connection or a dual mode device may receive an upgrade image from this higher speed interface directly to memory or to a host processor with storage space. The device must validate the image prior to use and execute the update when commanded.
- A device with other functionality or processors is not required to use the OTA software mechanism and validation for these other functions. For example, a gateway may have a large set of added functionality that is not subject to this specification.

## 6.4 Application Level Security

Each device is responsible to protect the resources it serves based on the policy for application level security. Each cluster has a defined set of security policies based on the needs of that cluster. In general these can be broken down as follows:

- Attribute access – the ability to access and manipulate attributes may or may not be protected by application level security based on the specified cluster.
- Commands – commands may or may not be protected by application level security based on the specified cluster.
- Reporting – The reporting of attributes (and configuration of reporting) security policies follows the security policy for attribute access above. If application level security is required, then reporting is also only done with application level security.
- Bindings – The use of binding security policy follows the security policy for what is being done with the binding. If attribute reporting is done then the binding security policy follows the policy for attributes. If a command is sent based on the binding then the security policy for commands SHALL be followed.

### 6.4.1 Using Application Layer Security

When Application layer security is required, devices SHALL not accept transactions without a DTLS session established between the devices. Devices SHALL use access control for protected resources as needed (refer to Section 6.5). Once a DTLS session is established devices are not required to keep the connection alive but SHOULD maintain session tokens to allow quicker reestablishment of the session.

Application layer security can be setup using one of the following methods:

If the devices share a common manufacturing certificate they can use this certificate to establish the DTLS session.

A DTLS session may be established using raw public key, an operational certificate or a preshared key but the following SHALL be done.

- The device must have a secure session established with a trusted device such as the device that commissioned it on the network.
- The trusted device can provide the UID of the device that can be trusted in the case of raw public key. The public key of that device can then be requested from it or the trusted device can provide it.
- The trusted device can provide an operational certificate to devices that are to establish a connection.
- The trusted device can provide the preshared key to the devices that are to establish a connection.

The device is responsible to maintain the pairing of security material for each device it is to establish a DTLS session with. For example, for each binding requiring application level security it must maintain the security material required to establish that session. It is sufficient to retain the public key for the device or even a hash of the public key to ensure the proper session is established.

The device SHALL retain the sequence number for the current session and SHALL use this to prevent replay attacks as described in [DTLS] 4.1.2.6. The minimum sliding window size for duplicate rejection SHALL be 64.

## 6.5 Cluster Security Requirements

For each cluster, Table 45 below indicates if access control is required for attributes, related metadata and for commands when accessed securely over unicast.

If access control is required for unicast communication, the device **MUST** protect interaction with the resource (i.e. attribute or command) when receiving or when transmitting to only those devices which have authorization for the specific resource according to 6.6. If devices are queried without access control over unicast when it is required, they **SHALL** return a security error. Devices **MAY** disable multicast access to resources, depending on security requirements, and only allow secure unicast communication using access control. If devices are queried over multicast when multicast access to the specific resource is disabled, they **SHALL NOT** process the request. Resources that require access control **SHALL NOT** be accessible using the broadcast group (2.5.2) when queried over multicast.

Cluster	Attribute	Commands	Notes
Basic	Required	Required	
Identify	--	--	
Groups	--	Required	
Scenes	--	Required	
On/Off	--	Required	
Level Control	--	Required	
Color control	--	Required	
Occupancy sensing	--	--	
OTA	Required	Required	Transfer of images requires application level security and images must be signed by manufacturer
Window Covering	--	Required	
Illuminance measurement	--	--	
Door Lock	Required	Required	
Shade Configuration	--	--	
Thermostat	--	Required	
Temp Measurement	--	Required	
Fan Control	--	Required	
IAS ACE	Required	Required	
IAS WD	Required	Required	
IAS Zones	Required	Required	

**Table 45. Cluster Security Requirements**

As new cluster support is added the security requirements must be included here.

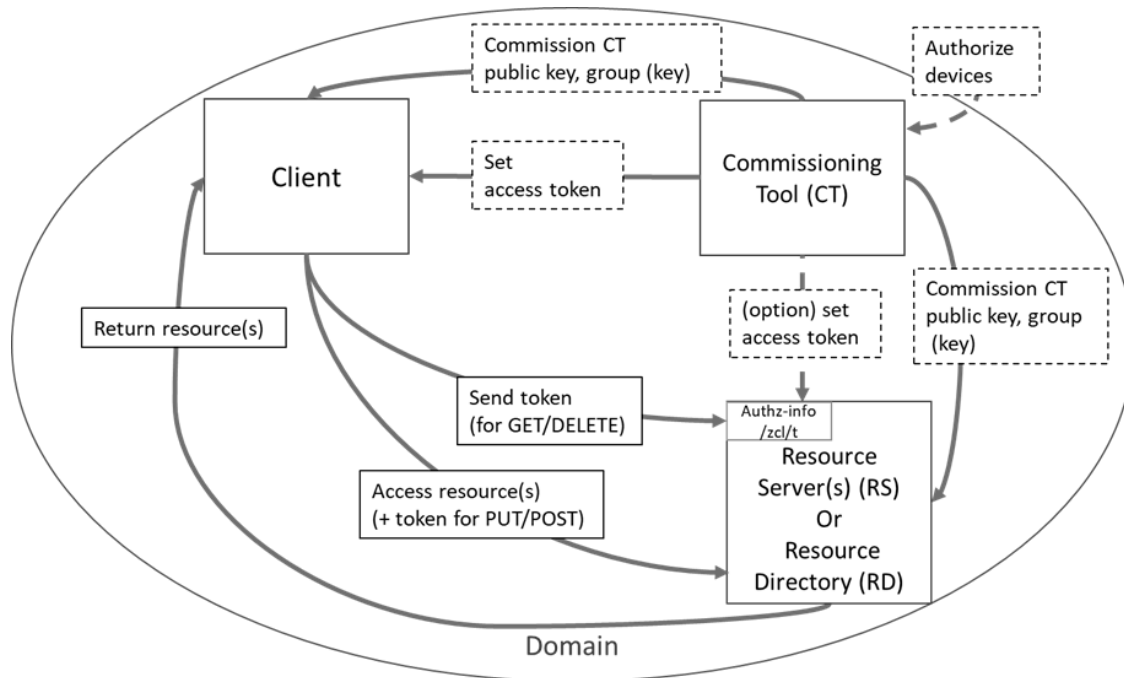


## 6.6 Access Control

Access control SHALL be used to provide authorization for access to protected resources. This paragraph describes authorization in ZCLIP, using some of the application layer specifics described in this document and based around [ACE] and [COSE]. Furthermore, the following constraints are made:

- Tokens SHALL have indefinite validity.
- Unicast communication SHALL be secured using DTLS with certificate security (client type confidential [OAUTH] 2.1). Additionally, RawPublicKey security MAY be used. If a certificate is presented and the signer is not recognized, it SHOULD be treated as a RawPublicKey. The use of PreSharedKey in combination with unicast SHALL not be supported.
- The UID, as described in section 2.7 SHALL be used as (client) key identifier ("kid") ([POPKS] 3.4).
- This method assumes that the client credentials (key identifier "kid") in the access token SHALL be cryptographically verified against the ZCLIP UID, as described in section 2.7, using DTLS for unicast. Protected resources SHALL never be accessible over unicast with no security or using a different key or with a different UID. Refer to ([OATMSC] 4.6.1).
- Tokens and related secret information SHALL never be sent in the clear.
- A limited number of groups of resources that can be protected is defined; individual resource protection is out-of-scope. Refer to the policies defined in section 6.4.
- Validation of client authorization SHALL be done on the resource server, using data provisioned by a commissioning tool, including validation of an access token signature using [COSE].
- Devices may be commissioned with access tokens via an out-of-band method ([ACE] 4), with some of the provisions in this document. By default all devices shall support the commissioning of access tokens as defined in section 6.2.7.
- For newer versions of this specification, new fields MAY be added to the payloads described here. Implementations SHALL for this reason ignore unknown fields in the payload. Any new fields SHALL be included in signature validation, to assure a correct signature computation.
- The CT SHALL be provided with device authorizations for protected resources out-of-band.
- The CT public key SHALL be provisioned on the devices via an out-of-band mechanism. The CT public key MAY already be available on devices depending on the provisioning of (operational) certificates described in sections 6.2.6 and 6.2.7.
- Other considerations in [OATMSC] SHOULD be followed where possible.

A diagram of the access control method described in the next sections is shown below:



**Figure 3. Overview of the Access Control Method**

## 6.6.1 Terminology

- Client: the ZCLIP client interface accessing a resource
- Resource server (RS): see Server - the ZCLIP server interface providing a resource
- Commissioning Tool (CT): the commissioning tool providing signed access tokens for a resource.
- Scope: the protected resources for which the client has access.
- Access token: a unique (per client/scope) signed token to an authorization for protected resources on a resource server.

## 6.6.2 Device Commissioning

The CT SHALL securely commission devices with the access token required for protected resources using this profile. The CT SHOULD commission the clients, but MAY instead commission the RS with the required access tokens. This section lists the parameters that the CT SHALL securely provide to the client, in CBOR format. By default, devices SHALL support the commissioning of access tokens as described in section 6.2.7.

Device SHOULD support the commissioning of multiple access tokens.

### 6.6.2.1 Client Parameters

The CT SHALL communicate the parameters listed in Table 43 to the client out of band. When the data is not communicated using a CBOR encoding, the Key and Encoding parameters MAY be ignored.

Name	Key	Type	Encoding	Default
Scope	12 ("scope")	Character string	Major type 3	NA
Profile	26 ("profile")	Character string	Major type 3	"coap_dtls"
Access token	19 ("access_token")	Character string (b64url)	Major type 3	NA
Confirmation claim	25 ("cnf")	Map	Major type 5	NA

**Table 46. Client Parameters**

The "scope" SHALL be set to the scope of groups of resources, device types and device unique identifiers the client is authorized access for, according to the following format:

```
"zcl (<scope>)"
with <scope> in the following format:
"[<RS UID> (..)] [d.<device id> (..)] [<cl>.<access group> (..)]" for device
resources
"[rd.r] [rd.r.<RDRES UID> (..)] [rd.r.d.< RDRES device id> (..)] [rd.r.<
RDRES cl> (..)]" for Resource Directory resources
```

Numeric values encoded in the scope SHALL be expressed in Base 16 (hexadecimal) using the same rules as for URI encoding as specified in 2.8.1.2. Where the "scope" SHALL specify at least one scope parameter. The scope parameters MAY be specified in any order.

Where a <cl>.<access group> MAY be included and is the resource group specifier. If included, the scope of access SHALL be valid only for the resource groups specified. It is defined as follows (6.4,6.5):

- The <cl> SHALL be set to a string with the Cluster Instance Identifier.
- The <access group> SHALL be set to a string with one of the following values:
  - "a" attribute access
  - "c" command access
  - "r" report access
  - "b" binding access

Where <RS UID> MAY be included and SHALL be set to the UID for the resource server, in the form "ni:///sha-256;..." (2.7). If included, scope of access SHALL be valid only for the resource servers with the specified UID.

Where <device id> MAY be included and SHALL be set to the Zigbee Device Type Identifier. If included, scope of access SHALL be valid only for the device types with the specified Device Type Identifier.

An example scope parameter that specifies access to reports and command resources of the basic cluster on all device types with id 106:

```
"zcl(d.106 s0.c s0.r)"
```

The value of "profile" SHALL be set to "coap\_dtls" [ACE] (11.6).

The confirmation claim "cnf" SHALL be set to a CBOR map including the parameters listed in Table 47.

Name	Key	Type	Encoding	Default
Key identifier	"kid"	Character string	Major type 3	NA

**Table 47. Confirmation Claim "cnf" Parameters**

The "kid" SHALL be set to a base64url value with contents "zcl" (b64'emNs').

### 6.6.2.2 Access Token Parameters

The value of "access\_token" SHALL be based on a CBOR web token [CWT], [CWTP]. The access token SHALL include the parameters listed in Table 48.

Name	Key	Type	Encoding	Default
Audience	3 ("aud")	Character string	Major type 3	NA
Confirmation claim	25 ("cnf")	Map	Major type 5	NA

**Table 48. Access Token Parameters**

The "aud" SHALL be set to the scope of groups of resources, device types or device unique identifiers for which the access\_token is valid, according to the same format and rules as defined for "scope" (6.6.2.1):

"zcl1([<RS UID> (...)] [d.<device id> (...)] [<cl>.<access group> (...)])"

The "aud" SHALL specify at least one scope parameter. The "aud" MAY include a <cl>.<access group> resource group specifier. The "aud" MAY include a <RS UID>. The "aud" MAY include a <device id>.

The confirmation claim "cnf" SHALL be set to a CBOR map including the parameters listed in Table 49.

Name	Key	Type	Encoding	Default
Key identifier	"kid"	Character string	Major type 3	NA

**Table 49. Access Token Confirmation Claim "cnf" Parameters**

The "kid" SHALL be set to the UID of the client for which the access\_token is valid, in the form "ni:///sha-256;..." (2.7).

The access token SHALL be signed by the CT using its private key, according to [COSE] (4.1), [CWTP] (7.1), [CWT] and using the algorithm ECDSA-with-SHA256 (6.2.5), tagged with CBOR tag "COSE\_Sign" ([IANA\_CBOR] 98). The header fields body\_protected, sign\_protected and external\_aad for the COSE Sig\_structure ([COSE] 4.4) SHALL be set to a zero length binary string. The access token SHALL be encoded in base64url format.

Example access\_token in binary format:

```
0xd862844040584ea203697a636c2873302e63291819a1636b696478396e693a2f2f2f7368612d3
235363b314a59587156536863434b74654a5f3161564d77786d466d58386a2d3650505470613864
504261386d776781834040584090c43fcb64afb225b701c0e4f4225cac5749c3a191982c752016b
5de9c3bfbdcdb71d7d71e756f1bf41abb637b0a09e0de4996606b89a27dbaf2495c19dcd8323
```

Example access\_token in base64url format:

```
"2GKEQEByTqIDaXpjbChzMC5jKRgZoWNraWR4OW5pOi8vL3NoYS0yNTY7MUUpZWJhFwU2hjQ0t0ZUpfMW
FWTXd4bUZtWDhqLTZQUFRwYThkUEJhOG13Z4GDQEBYQJDEP8tkr7IltwHA5PQiXKxXScOhkZgsdSAWt
d6cO_3L1x19cedW8b9Bq7Y3sKCeDeSZZga4mifbrySVwZ3NgyM"
```

## 6.6.3 Token resource

The URI Path

/zcl/t

2965 SHALL provide access to the authorization information (token) resource [ACE] of the device.

### 2966 6.6.3.1 Environment variables

2967 The environment variables that are used by the RS in handling the token requests are defined in Table 50.  
2968 These assume that the RS has a clock.

Name	Type	Description
current_token_time	UTCtime	Time at which the token POST request has been received by the RS
oldest_token_time	UTCtime	Time at which the oldest token request was successfully processed and stored
minimum_cache_time	UTCtime	2 * EXCHANGE_LIFETIME ([COAP])

2969 **Table 50 Environment variables**

### 2970 6.6.3.2 POST request

2971 To configure a token on a RS, a requesting device SHALL securely send a POST request to the token  
2972 resource. The payload SHALL be a CBOR map with the "access\_token" parameter according to Table 51,  
2973 encoded in base64url format. The request to the token resource by a requesting device to the RS SHALL  
2974 occur only using a CoAP over DTLS for unicast request.

### 2975 6.6.3.3 POST response

2976 If a requesting device sends a token resource request and does not connect securely with one of the  
2977 supported methods for unicast, RS SHALL return 4.00 bad request and further processing SHALL cease. If  
2978 a requesting device sends the token resource request over multicast, further processing SHALL cease. In  
2979 case of a successful POST request, the RS SHALL update current\_token\_time with the current UTC time,  
2980 unless the RS does not have a clock.

2981 The RS SHALL store the access token provided by the requesting device. The RS SHALL store the access  
2982 token for at least minimum\_cache\_time. The RS SHALL be able to store at least one access token. If the  
2983 RS does not have a clock, the RS SHALL be able to store at least two access tokens and SHALL mark the  
2984 oldest (first received) token "used". In case the RS temporarily cannot store more tokens, the following  
2985 mechanism SHALL be used:

2986 The RS SHALL try to overwrite a token that was marked "used" as defined in 6.6.4. A device MAY  
2987 support the caching of multiple access tokens. In case this still fails, the RS SHALL try to overwrite a token  
2988 for which the minimum\_cache\_time has exceeded, according to a first-in-first-out (FIFO) method. In case  
2989 the RS still cannot store the access token, it SHALL return a 5.03 Service Unavailable response, and set the  
2990 Max-Age option to the retry\_time, which is the number of seconds after which a token is guaranteed to be  
2991 allowed to be stored again, defined as:

2992 
$$\text{retry\_time} = \text{oldest\_token\_time} + \text{minimum\_cache\_time} - \text{current\_token\_time}$$

2993 The RS SHALL verify the access token signature using the stored public key of the commissioning tool  
2994 according to [COSE] (4.4), [CWT] (7.2), [CWTP] and using the algorithm ECDSA-with-SHA256 (6.2.5).  
2995 In case the signature verification fails, the RS SHALL remove the token and SHALL return 4.01  
2996 unauthorized and further processing SHALL cease. The RS SHALL verify the "aud" scope of the access  
2997 token against the available protected resources on the RS according to the rules defined for the scope of  
2998 access in 6.6.2.1. In case the verification fails, the RS SHALL remove the token and SHALL return a 4.03

2999 forbidden response and further processing SHALL cease. In case the access token with the same  
 3000 parameters already exists on the RS, the RS SHALL only store one copy of the token.

3001 The RS SHALL, in case no error occurred, return a POST response message containing a CoAP response  
 3002 code of 2.01 Created.

#### 3003 6.6.4 Protected Resource Request

3004 A request to a protected resource is defined as any GET/PUT/POST/DELETE CoAP request to a resource  
 3005 as defined in Section 3 with the restrictions described in Section 6.5.

3006 The client SHALL determine if a resource is protected using the parameters specified in 6.6.2.1, and  
 3007 whether access to the resource was previously authorized.

3008 For any unicast request to a protected resource that was not previously authorized and for which a payload  
 3009 is allowed according to [CoAP] (i.e. POST/PUT), the client SHALL add a CBOR map with the  
 3010 "access\_token" parameter to the CBOR encoded payload according to Table 51, encoded in base64url  
 3011 format. Any unicast request to a protected resource for which a payload is not allowed according to [CoAP]  
 3012 (i.e. GET/DELETE) and for which the client was not previously authorized, SHALL be directly preceded  
 3013 by a POST of the token information to the token resource according to 6.6.3. The key for the CBOR map  
 3014 when embedded inside a map SHALL be set to "at". If no payload is present, the client SHALL create a  
 3015 CBOR map with the "access\_token" parameter. The "access\_token" SHALL be tagged with CBOR tag  
 3016 "COSE\_Sign".

3017 For unicast requests, in case the RS already was provided with a valid access token for the client, the client  
 3018 SHOULD leave out the access\_token parameter from requests for the same RS with the same public or  
 3019 shared key.

3020 The request to the protected resource by the client to the RS SHALL for unicast occur only using CoAP  
 3021 over DTLS.

Name	Key	Type	Encoding	Default
Access token	19 ("access_token")	Character string (b64url)	Major type 3	NA

3022 **Table 51. Client Request Access Token Parameters**

#### 3023 6.6.5 Protected Resource Response

3024 If the client requests a protected resource and does not connect with one of the supported methods for  
 3025 unicast, RS SHALL return 4.00 bad request and further processing SHALL cease. The RS SHALL, for  
 3026 unicast communication, store the UID value of the client calculated from the client public key according to  
 3027 2.7.1, and use this as Key Identifier ("kid").

3028 The RS SHALL, for unicast connections using POST or PUT, store the access token provided by the client.  
 3029 The RS SHALL allow a client that was previously authorized for access to request a protected resource  
 3030 over unicast without including the access token again if the connection uses the same client public key  
 3031 ("kid"). In case an access token was not provided and the client was not previously authorized, the RS  
 3032 SHALL retrieve the access token that was stored previously for this client according to 6.6.3 and process  
 3033 the token as if it was sent by the client. The RS SHALL mark a retrieved access token as "used". If the  
 3034 client does not include an access token for a protected resource, the RS SHALL return 4.01 unauthorized,  
 3035 and further processing SHALL cease. If the client was not previously authorized for access for unicast  
 3036 requests, the RS SHALL return 4.01 unauthorized and further processing SHALL cease.

3037 For unicast connections that contain an access token payload, the RS SHALL verify the access token  
 3038 signature using the stored public key of the commissioning tool according to [COSE] (4.4), [CWT] (7.2),

3039 [CWTP] and using the algorithm ECDSA-with-SHA256 (6.2.5). In case the signature verification fails, the  
3040 RS SHALL return 4.01 unauthorized and further processing SHALL cease.

3041 For unicast connections, the RS SHALL verify the "kid" of the client against the "kid" value of the access  
3042 token. In case the "kid" is not found or different from the client "kid", the RS SHALL return a 4.03  
3043 forbidden response and further processing SHALL cease.

3044 The RS SHALL, for unicast connections, verify the "aud" scope of the access token against the request  
3045 made by the client according to the rules defined for the scope of access in 6.6.2.1. In case the verification  
3046 fails, the RS SHALL return a 4.03 forbidden response and further processing SHALL cease.

3047 In case the client is authorized to access the resource, the RS SHALL allow the client to access the  
3048 protected resource according to Section 3.

3049

## 7 OTA Bootload

3050

3051 All device SHALL support OTA Upgrading. The mechanism for Over-The\_Air (OTA) bootload is defined  
3052 in [ZCLIPOTA].

3053



## 8 ZCLIP Cluster Translation Rules

This chapter specifies the rules used to translate a cluster definition specified in [ZCL] into a form suitable for ZCLIP.

### 8.1 Cluster Attributes and Metadata

Attribute and metadata values SHALL be encoded according to the ZCL-to-ZCLIP type mapping specified in Section 2.8.2.2.

Attributes and metadata SHALL be accessed in a standard manner across clusters via the Endpoint and Group Attribute URI resources. No cluster-specific treatment of attributes is specified or needed.

### 8.2 Cluster Commands

ZCL defines commands as either generated or received. In both cases, an examination of the transaction pattern is necessary to determine if a command falls into the ZCL Request command or ZCL Response command categories. Frequently the name will indicate if the command is a request or response message. ZCL Request messages SHALL be exposed as a command instance in the relevant command collection for a cluster instance. Messages that are not classified as ZCL Request messages SHALL NOT be exposed as a command instance in command collection for the cluster instance.

#### 8.2.1 ZCL Request Command

The ZCLIP form of a ZCL request command SHALL be a CoAP POST request directed to the Endpoint Command resource (/zcl/e/<eid>/<cl>/<cid>) or to the Group Command resource (/zcl/g/<gid>/<cl>/<cid>), where:

<cl> is the cluster instance identifier, comprised of the cluster side, optional manufacturer extension, and cluster ID, as specified in Section 2.1.4.

<cid> is the cluster command identifier for the cluster command to be executed.

ZCL cluster request command payload parameters SHALL be encoded into the payload of the CoAP request message as described in Section 8.2.3.

#### 8.2.2 ZCL Response Command

The ZCLIP form of a ZCL response command SHALL be the CoAP piggyback or separate response message to a received CoAP request representing a ZCL request command.

The success or failure of the command is indicated in the CoAP response code. The code SHALL be 2.04 Changed for successful command execution, and a 4.xx or 5.xx error code as appropriate to indicate a command execution failure.

ZCL cluster response command payload parameters for a response SHALL be encoded into the payload of the CoAP response message as described in Section 8.2.3.

#### 8.2.3 Payload Translation Rules

This section describes how to render the [ZCL] definition of a cluster command payload into ZCLIP form.

### 8.2.3.1 ZCL Payload Elements

For purpose of ensuing specification, the following terminology is defined to refer to the elements in a ZCL payload.

- Scalar** Single data element, other than a Sizer or Typer, that contains a value of a specified ZCL data type.
- Record** Contiguous sequence of elements that collectively represent a unit of information. Each element in a record may be a Scalar, a Record, or a Collection. For example, an attribute record might contain an attribute ID, an attribute type, and an attribute value.
- Collection** Contiguous sequence of a variable number of semantically homogeneous Scalars or Records; for example, a sequence of attribute IDs, or a sequence of attribute records. As defined here, a Collection is always accompanied by a related Sizer element in the ZCL payload that specifies the number of elements in the Collection.
- Sizer** Integer that represents a count, length, or size; the purpose of which is to support parsing of binary ZCL messages by directly indicating the number or size of subsequent information in the ZCL payload.
- Typer** Enumerated ZCL type; the purpose of which is to support parsing of binary ZCL messages by indirectly indicating the size of subsequent information

### 8.2.3.2 Empty Payload

If [ZCL] specifies no payload for the message, the ZCLIP message payload SHALL be empty.

### 8.2.3.3 Nonempty Payload

If [ZCL] specifies a payload for the message, the defined ZCL payload SHALL be treated as a Record and encoded in the ZCLIP payload as specified in Section 8.2.3.4.

The decomposition and encoding rules SHALL be applied recursively to nested Collections and Records until a Scalar representation of data is reached and no further decomposition is possible.

### 8.2.3.4 ZCLIP Encoding of a Record

A Record SHALL be encoded as a map of entries representing the elements of the Record.

Examine the payload structure as specified in [ZCL] and apply the following decomposition rules to determine the set of items included in the map, and their map key numbering.

1. Identify and classify Record elements at the coarsest level of element granularity, as follows:
  - a. From coarsest to finest, the classification SHALL be:
    - i. Collection
    - ii. Record that is not an element of a Collection
    - iii. Scalar that is not an element of a Collection or a Record
    - iv. Sizer or Typer that is not an element of a Record
  - b. A fixed number of sequential Scalars or Records SHALL be classified as separate elements; they SHALL NOT be classified as a Collection (there is no associated Sizer or Typer).

- 3111 c. The finest granularity SHALL be one octet. Integer, enumeration, and Boolean values  
 3112 defined by [ZCL] to be contained in bit fields smaller than one octet SHALL NOT be  
 3113 represented as separate elements.
- 3114 d. The granularity of payload elements as defined in [ZCL] SHALL be preserved, even  
 3115 though it may not result in the most natural or efficient ZCLIP encoding. For example, a  
 3116 definition of sixteen sequential ZCL map16 fields, which collectively represent a 256-bit  
 3117 bitmap, would be classified and encoded as sixteen separate elements; and not classified  
 3118 and encoded as a single 256-bit element.
- 3119 2. Sizer and Typer elements SHALL be omitted from further consideration. The information  
 3120 represented by a Sizer/Typer is captured in the encoding of the element for which the Sizer/Typer  
 3121 specifies the size. The Sizer/Typer SHALL NOT be explicitly and redundantly represented in the  
 3122 ZCLIP payload.
- 3123 3. For the remaining identified elements, sequential map key numbers starting from zero SHALL be  
 3124 assigned in the order the elements appear in the ZCL payload. An element that [ZCL] specifies as  
 3125 optionally or conditionally present in the ZCL payload SHALL be assigned its own distinct  
 3126 number. Thus all elements that may be present in a Record SHALL have a unique number that  
 3127 unambiguously represents it, regardless of how many of the elements are actually present in a  
 3128 particular instance of that Record.
- 3129 An entry for each element present in an instance of the Record SHALL be inserted into the map.
- 3130 The entry key SHALL be the element's assigned key number. The set of map key values may be sparse if  
 3131 one or more optional or conditional elements is not present in the Record instance.
- 3132 The entry value SHALL be the ZCLIP encoding of the element according to its element type (Scalar,  
 3133 Record, Collection).

### 3134 8.2.3.5 ZCLIP Encoding of a Scalar

3135 A Scalar SHALL be encoded according to the ZCL-to-ZCLIP data type mapping specified in Section  
 3136 2.8.2.2.

### 3137 8.2.3.6 ZCLIP Encoding of a Collection

- 3138 A Collection SHALL be encoded as a map of entries representing the elements of the Collection.
- 3139 An entry for each element of the Collection SHALL be inserted into the map.
- 3140 The entry key SHALL be an integer in range 0 to N-1 according to the element's relative position within  
 3141 the Collection, where N is the number of elements in the collection as specified by the Collection's  
 3142 associated Sizer field in the ZCL payload.
- 3143 The entry value SHALL be the ZCLIP encoding of the Collection element according to its element type  
 3144 (Scalar, Record, Collection).

## 3145 8.2.4 ZCL Cluster-Specific Clarifications

3146 Where the procedures of this chapter fall short of unambiguously resolving the ZCLIP representation of a  
 3147 particular cluster, clarification will be added in Annex E.

3148

3149

## Annex A CoAP Response Codes

The following table summarizes the CoAP response codes returned for requests issued with the various method types to the various URI resource types. Response codes 2.xx indicate success, and for POST specify the resulting change. For certain combinations, the method is not supported for the resource type, and the only response code is either 4.05 Method Not Allowed or possibly 4.04 Not Found. For combinations for which the method is supported for the resource type, both success and error response codes are shown. Payload content is described in the specification for the particular resource or error type.

All combinations potentially can return 4.00 Bad Request (an error is detected with the contents of the request), 5.00 Internal Server Error (an unexpected condition prevented the server from processing the request) or 5.03 Service Unavailable (the server cannot process the request because it is temporarily overloaded or down for maintenance).

TARGET	METHOD			
	GET	PUT	POST	DELETE
Collection of resources	2.05 Content	4.05 Method Not Allowed	2.01 Created 2.04 Changed  4.05 Method Not Allowed if collection does not allow modifications	4.05 Method Not Allowed
Accessible Resource instance	2.05 Content	2.04 Changed	4.05 Method Not Allowed	2.02 Deleted  4.05 Method Not Allowed if deletion is not allowed
Action-only resource	4.05 Method Not Allowed	4.05 Method Not Allowed	2.01 Created 2.02 Deleted 2.04 Changed	4.05 Method Not Allowed
Nonexistent Resource	4.04 Not Found	4.04 Not Found or MAY return  4.05 Method Not Allowed	4.04 Not Found or MAY return  4.05 Method Not Allowed	2.02 Deleted or MAY return  4.04 Not Found  4.05 Method Not Allowed

**Table 52: Coap Response Codes**

## Annex B ZCL General Request Command Mapping

The table below illustrates the mapping of ZCL General Request Commands to related REST method invocations on ZCLIP URI resources. All examples are expressed in terms of Endpoint URIs (/zcl/e/<eid>/...). Some may also be supported via group access (e.g. URI /zcl/g/<gid>/...).

ZCL General Command	ZCLIP Method and URI Resource
Read Attributes	GET /zcl/e/<eid>/<cl>/a?f=<attr_filter> GET /zcl/e/<eid>/<cl>/a/<aid>
Write Attributes	POST /zcl/e/<eid>/<cl>/a PUT /zcl/e/<eid>/<cl>/a/<aid>
Write Attributes Undivided	POST /zcl/e/<eid>/<cl>/a?u
Write Attributes No Response	POST /zcl/e/<eid>/<cl>/a (no response returned)
Configure Reporting	POST /zcl/e/<eid>/<cl>/r PUT /zcl/e/<eid>/<cl>/r/<rid> DELETE /zcl/e/<eid>/<cl>/r/<rid>
Read Reporting Configuration	GET /zcl/e/<eid>/<cl>/r GET /zcl/e/<eid>/<cl>/r/<rid>
Report Attributes	POST /zcl/e/<eid>/<cl>/n
Discover Attributes	GET /zcl/e/<eid>/<cl>/a?meta=\$base&f=*
Read Attributes Structured	(not supported)
Write Attributes Structured	(not supported)
Discover Commands Received	GET /zcl/e/<eid>/<cl>/c
Discover Commands Generated	(not supported)
Discover Attributes Extended	GET /zcl/e/<eid>/<cl>/a?meta=\$base,\$acc&f=*

**Table 53. ZCL General Request Mapping**

## Annex C Reference Device for Test

The ZCLIP Base Device specification uses a given set of sample devices for the purposes of illustrating examples in the text. The example devices consist of a ZCLIP Light and a Switch. In the text of the Base Device Specification and also in the ZCLIP Test Specification these devices are used as examples and are referenced as ZCLIP Light and ZCLIP Switch. The example devices used are configured as follows:

### C.1 ZCLIP Light

The ZCLIP Light used in examples in the ZCLIP Base Device specification has two endpoints. The endpoints described in the ZCLIP Light are as follows:

#### Endpoint: 1

##### Device ID: 0x0102 (Color Dimmable Light)

Cluster Id	Cluster Side(s)	Cluster Purpose	Attributes
0x0	Server	Basic	Required + optional attribute 0x0001, Application Version
0x3	Server	Identify	Required
0x4	Server	Groups	Required
0x5	Server	Scenes	Required
0x6	Server	On/Off	Required
0x8	Server	Level Control	Required
0x300	Server	Color Control	Required

**Table 54: ZCLIP Light Endpoint 1 Clusters**

#### Endpoint: 2

##### Device ID: 0x0107 (Occupancy Sensor)

Cluster Id	Cluster Side(s)	Cluster Purpose	Attributes
0x0	Server	Basic	Required + optional attribute 0x0001, Application Version
0x3	Client / Server	Identify	Required
0x4	Server	Groups	Required
0x406	Server	Occupancy Sensing	Required

**Table 55: ZCLIP Light Endpoint 2 Clusters**

## C.2 ZCLIP Switch

The ZCLIP Color Dimmable Light Switch used in examples in the ZCLIP Base Device specification has two endpoints each of which implement all clusters necessary for a basic Zigbee color controllable dimmer switch. The Clusters implemented are as follows:

### Endpoint: 1

#### Device ID: 0x0105 (Color Dimmer Switch)

Cluster Id	Cluster Side(s)	Cluster Purpose	Attributes
0x0	Server	Basic	Required + optional attribute 0x0001, Application Version
0x3	Client / Server	Identify	Required
0x6	Client	On/Off	Required
0x8	Client	Level Control	Required
0x300	Client	Color Control	Required

**Table 56: ZCLIP Switch Endpoint 1 Clusters**

### Endpoint: 2

#### Device ID: 0x0105 (Color Dimmer Switch)

Cluster Id	Cluster Side(s)	Cluster Purpose	Attributes
0x0	Server	Basic	Required + optional attribute 0x0001, Application Version
0x3	Client / Server	Identify	Required
0x6	Client	On/Off	Required
0x8	Client	Level Control	Required
0x300	Client	Color Control	Required

**Table 57: ZCLIP Switch Endpoint 2 Clusters**

## Annex D Examples

### D.1 ZCLIP URI Examples

No	Example	Description
1	/zcl/e/1/s1/a?f=0	Endpoint 1, server cluster 1, attribute 0
2	/zcl/e/1/s1/a/0	Endpoint 1, server cluster 1, attribute 0
3	/zcl/g/1/s1/a?f=0	Group 1, server cluster 1, attribute 0
4	/zcl/g/1/s1/a/0	Group 1, server cluster 1, attribute 0
5	/zcl/e/1/s190a_fc0d/c/a	Endpoint 1, manufacturer-defined (0x190a) server cluster 0xfc0d (64525), command 10
6	/zcl/e/3/s190a_1c4/c/2	Endpoint 3, manufacturer-defined (0x190a) server cluster 0x1c4 (452), manufacturer-defined (0x190a) command 2
7	/zcl/m/c/0	EZ-Mode advertisement

Table 58. URI Examples

### D.2 Attribute URI Query Examples

Cluster has attributes [1, 3, 4, d, e]

URI	Comment
/zcl/e/1/s6/a	Response returns [1,3,4,d,e]
/zcl/e/1/s6/a?f=0,4+2	Response returns {4: <value>, d: <value>}
/zcl/e/1/s6/a?f=0-2	Response returns {1: <value>}
/zcl/e/1/s6/a?u	Indicates the associated multiple update must be processed undivided ("all or nothing")

Table 59. Attribute Filtering URI Examples

### D.3 Discovery Query Examples

#### D.3.1 Query to CoRE Root

Unicast query to the .well-known/core discovery resource.

Request:



3211 GET coaps://[2002:8290:4eed::8290:4eed]/.well-known/core

3212 Response:

3213 2.05 Content (Content-Format: application/link-format+cbor)  
 3214 [{1: "coaps://[2002:8290:4eed::8290:4eed]/zcl", 9: "urn:zcl", 10:  
 3215 "urn:zcl:v1"},  
 3216 {1: "coaps://[2002:8290:4eed::8290:4eed]/temperature", 9:  
 3217 "urn:xyz:phys:temp-c"}]

3218

3219 The result indicates that the targeted device contains a ZCLIP entry point and that it also hosts a resource or  
 3220 service ‘temperature’ which is defined by another protocol, not Zigbee related. The Zigbee entry-point  
 3221 resource /zcl has the resource type rt=urn:zcl and if=urn:zcl:v1 which describes the ZCLIP protocol  
 3222 revision (as explained in 4.3.3.2).

### 3223 D.3.2 Query using the ‘rt’ attribute

3224 Multicast query using the ‘rt’ attribute to discover Zigbee endpoints implementing the client side of the  
 3225 on/off cluster (ClusterID 0x0006).

3226 Request:

3227 GET coap://[ff03::fd]/.well-known/core?rt=urn:zcl:c.6.c

3228 Response 1:

3229 2.05 Content (Content-Format: application/link-format+cbor)  
 3230 [{1: "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/c6", 9: "urn:zcl:c.6.c",  
 3231 10: "urn:zcl:c.v1", "ze": "urn:zcl:d.103.1"},  
 3232 {1: "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/c6", 9: "urn:zcl:c.6.c",  
 3233 10: "urn:zcl:c.v1", "ze": "urn:zcl:d.103.25"}]

3234 Response 2:

3235 2.05 Content (Content-Format: application/link-format+cbor)  
 3236  
 3237 [{1: "coap://[2002:8290:4eed::ae0d:2bed]/zcl/e/7/c6", 9: "urn:zcl:c.6.c",  
 3238 10: "urn:zcl:c.v1", "ze": "urn:zcl:d.104.7"},  
 3239 {1: "coap://[2002:8290:4eed::ae0d:2bed]/zcl/e/4/c6", 9: "urn:zcl:c.6.c",  
 3240 10: "urn:zcl:c.v1", "ze": "urn:zcl:d.104.4"}]

3241

3242

3243 This shows there are two Zigbee endpoints present on the responding node (endpoint 1 and endpoint 25),  
 3244 both having the client side of the on/off cluster implemented.

3245 In a similar way, a request to discover all Zigbee endpoints implementing the server side of the on/off  
 3246 cluster will have the following syntax:

3247 GET coap://[ff03::fd]/.well-known/core?rt= urn:zcl:c.6.s

3248 If a client is interested to discover all Zigbee endpoints implementing the on/off cluster, independent from  
 3249 the server/client side, the wildcard character can be used as shown in the following example:

3250 GET coap://[ff03::fd]/.well-known/core?rt=urn:zcl:c.6.\*

### 3251 D.3.3 Query using the ‘if’ Attribute

3252 Multicast query using the ‘if’ attribute to discover all Zigbee entry points which implement a specific  
 3253 version of ZCLIP.

3254 Request:

3255 GET coap://[ff03::fd]/.well-known/core?rt=urn:zcl&if=urn:zcl:v1

3256 Response 1:

3257 2.05 Content (Content-Format: application/link-format+cbor)  
 3258 [{1: "coap://[2002:8290:4eed::8290:4eed]/zcl", 9: "urn:zcl", 10:  
 3259 "urn:zcl:v1", "ep"="ni:/// sha-  
 3260 256;NvbKUrHKwCpPamtO80GvjpIoHyUBWwgfhoOlEpSlbU"}]

3261

3262 Response 2:

3263 2.05 Content (Content-Format: application/link-format+cbor)  
 3264 [{1: "coap://[2002:8290:4eed::ae0d:48c3]/zcl", 9: "urn:zcl", 10:  
 3265 "urn:zcl:v1", "ep"="ni:/// sha-  
 3266 256;UvbKrrHKwpaamtO80GvjpIoHyUBWwgfhoOfEpElbS"}]

3267 This shows there are two Zigbee entry points present, both having the ZCLIP protocol version requested.

3268 Multicast query using the ‘if’ attribute to discover all the Zigbee endpoints implementing version 5 of the  
 3269 on/off Cluster:

3270 Request:

3271 GET coap://[ff03::fd]/.well-known/core?rt=urn:zcl:c.6.\*&if=urn:zcl:c.v5

3272 Response 1:

3273 2.05 Content (Content-Format: application/link-format+cbor)  
 3274 [{1: "coap://[2002:8290:4eed::8290:4eed]/zcl/e/24/c6", 9:  
 3275 "urn:zcl:c.6.c", "ze": "urn:zcl:d.107.24", 10: "urn:zcl:c.v5"}]

3276

3277 Response 2:

3278 2.05 Content (Content-Format: application/link-format+cbor)  
 3279 [{1: "coap://[2002:8290:4eed::ae0d:48c3]/zcl/e/3/c6", 9: "urn:zcl:c.6.s",  
 3280 "ze": "urn:zcl:d.107.3", 10: "urn:zcl:c.v5"}]

3281 Multicast query using the ‘if’ attribute to discover all the Dimmable lights (DeviceID 0x0101) which  
 3282 implement a specific version of the client side of the Occupancy sensing Cluster (Cluster ID 0x406). (Note:  
 3283 The Occupancy sensing Cluster is an optional cluster for dimmable light devices).

3284 Request:

3285 GET coap://[ff0x::fd]/.well-  
 3286 known/core?ze=urn:zcl:d.101.\*&rt=urn:zcl:c.406.c&if=urn:zcl:c.v3

3287 Response 1:

3288 2.05 Content (Content-Format: application/link-format+cbor)  
 3289 [{1: "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/c406", 9:  
 3290 "urn:zcl:c.406.c", "ze": "urn:zcl:d.101.1", 10: "urn:zcl:c.v3"}]

3291

3292 Response 2:

3293 2.05 Content (Content-Format: application/link-format+cbor)  
 3294 [{1: "coap://[2002:8290:4eed::ae0d:48c3]/zcl/e/5/c406", 9:  
 3295 "urn:zcl:c.406.c", "ze": "urn:zcl:d.101.5", 10: "urn:zcl:c.v3"}]

3296

### 3297 D.3.4 Query using the ‘ze’ Attribute

3298 Multicast query using ‘ze’ attribute to request all the Zigbee endpoints matching DeviceID 0x107  
 3299 (Occupancy sensors).

3300 Request:

3301 GET coap://[ff05::fd]/.well-known/core?ze=urn:zcl:d.107.\*

3302 Response 1:

3303 2.05 Content (Content-Format: application/link-format+cbor)  
 3304 [{1: "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1", 10: "urn:zcl:d.v2",  
 3305 "ze": "urn:zcl:d.107.1"}, {1:  
 3306 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/s0", 9: "urn:zcl:c.0.s",  
 3307 "ze": "urn:zcl:d.107.1", 10: "urn:zcl:c.v3"}, {1:  
 3308 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/s3", 9: "urn:zcl:c.3.s",  
 3309 "ze": "urn:zcl:d.107.1", 10: "urn:zcl:c.v3"}, {1:  
 3310 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/s406", 9: "urn:zcl:c.406.s",  
 3311 "ze": "urn:zcl:d.107.1", 10: "urn:zcl:c.v3"}, {1:  
 3312 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/1/c3", 9: "urn:zcl:c.3.c",  
 3313 "ze": "urn:zcl:d.107.1", 10: "urn:zcl:c.v3"}]

3314 Response 2:

3315 2.05 Content (Content-Format: application/link-format+cbor)  
 3316 [{1: "coap://[2002:8290:4eed::8290:2bed]/zcl/e/4", 10: "urn:zcl:d.v2",  
 3317 "ze": "urn:zcl:d.107.4"}, {1:  
 3318 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/4/s0", 9: "urn:zcl:c.0.s",  
 3319 "ze": "urn:zcl:d.107.4", 10: "urn:zcl:c.v3"}, {1:  
 3320 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/4/s3", 9: "urn:zcl:c.3.s",  
 3321 "ze": "urn:zcl:d.107.4", 10: "urn:zcl:c.v3"}, {1:  
 3322 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/4/s406", 9: "urn:zcl:c.406.s",  
 3323 "ze": "urn:zcl:d.107.4", 10: "urn:zcl:c.v3"}, {1:  
 3324 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/4/c3", 9: "urn:zcl:c.3.c",  
 3325 "ze": "urn:zcl:d.107.4", 10: "urn:zcl:c.v3"}]

3326

3327 Response 3:

3328 2.05 Content (Content-Format: application/link-format+cbor)  
 3329 [{1: "coap://[2002:8290:4eed::ae0d:48c3]/zcl/e/25", 10: "urn:zcl:d.v2",  
 3330 "ze": "urn:zcl:d.107.25"}, {1:  
 3331 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/s0", 9: "urn:zcl:c.0.s",  
 3332 "ze": "urn:zcl:d.107.25", 10: "urn:zcl:c.v3"}, {1:  
 3333 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/s3", 9: "urn:zcl:c.3.s",  
 3334 "ze": "urn:zcl:d.107.25", 10: "urn:zcl:c.v3"}, {1:  
 3335 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/s406", 9: "urn:zcl:c.406.s",  
 3336 "ze": "urn:zcl:d.107.25", 10: "urn:zcl:c.v3"}, {1:  
 3337 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/c3", 9: "urn:zcl:c.3.c",  
 3338 "ze": "urn:zcl:d.107.25", 10: "urn:zcl:c.v3"}, {1:  
 3339 "coap://[2002:8290:4eed::8290:4eed]/zcl/e/25/c4", 9: "urn:zcl:c.4.c",  
 3340 "ze": "urn:zcl:d.107.25", 10: "urn:zcl:c.v3"}]

3341

### 3342 D.3.5 Query for Active Clusters on a Zigbee Endpoint

3343 Unicast query using 'rt' attribute to discover all the active Zigbee clusters implemented on a ZCLIP node:

3344 GET coaps://[2002:8290:4eed::8290:4eed]/.well-known/core?rt=urn:zcl:c.\*

3345 Response:

3346 2.05 Content (Content-Format: application/link-format+cbor)  
 3347 [{1: "coaps://[2002:8290:4eed::8290:4eed]/zcl/e/4/c0", 9:  
 3348 "urn:zcl:c.0.c", "ze": "urn:zcl:d.107.4"},  
 3349 {1: "coaps://[2002:8290:4eed::8290:4eed]/zcl/e/5/c3", 9: "urn:zcl:c.3.c",  
 3350 10: "urn:zcl:c.v3", "ze": "urn:zcl:d.107.5"},

3351 {1: "coaps://[2002:8290:4eed::8290:4eed]/zcl/e/21/c406", 9:  
3352 "urn:zcl:c.406.c", 10: "urn:zcl:c.v3", "ze": "urn:zcl:d.107.21"}]

### 3353 D.3.6 Query using the 'ep' Attribute

3354 Multicast query request using 'ep' attribute to discover a ZCLIP device which matches a specific UID.

3355 This discovery is useful after events like IPv6 prefix change, IPv6 address change, or relocation of a ZCLIP  
3356 device to another network segment.

3357 Request:

3358 GET coap://[ff0x::fd]/.well-known/core?ep=ni:///sha-256;<hash prefix>\*

3359 Response:

3360 2.05 Content (Content-Format: application/link-format+cbor)  
3361 [{1: "coap://[nodeIPv6Addr]/zcl", 9: "urn:zcl", 10: "urn:zcl:v1,  
3362 "ep"="ni:///sha-256;<hash value>"}]

3363

## Annex E Cluster Implementations

General rules for translating ZCL cluster functionality into ZCLIP transactions, for ZCL Version 6 and above, are provided in the text of this specification. For most clusters, application of the rules will produce a unique ZCLIP implementation. For clusters that have special characteristics (such as layering violations) that may make the ZCLIP implementation ambiguous, this Annex provides clarification to resolve the ambiguity and ensure interoperable implementations.

### E.1 Groups Cluster

Group operations are implemented leveraging the ZCLIP implementation of the group cluster commands, in accordance with [ZCL] Section 3.6, which include:

- Add group membership to an endpoint
- View group membership of an endpoint
- Get group memberships of an endpoint
- Remove all group memberships of an endpoint
- Add group membership to an endpoint, on condition that it is identifying itself.

Group cluster commands are available at URI-Path `/zcl/e/<eid>/s4/c/<cid>` with `<eid>` representing the target endpoint ID, `s4` representing the server side of the Groups cluster, and `<cid>` representing the command ID of the specific command considered.

#### E.1.1 Add Group

To add group membership to an endpoint of a target device, the sending device SHALL send a POST request to the URI-Path `/zcl/e/<eid>/s4/c/0`. The payload SHALL contain a map of command parameters with the key/value pairs shown in Table 60.

Name	Key	Type	Encoding
Group ID	0	16-bit unsigned integer	Major type 0
Group Name	1	Character String	Major type 3

**Table 60. Add Group Command Parameters**

Group ID represents the identifier of the group requested to be added with this command.

Group Name represents a 0 to 16 character text identifier of the group requested to be added with this command.

The device MAY include the Address Assignment Mode, IPv6 Address and Group Port extensions. If these are not included, the default value specified in the extensions field SHALL be assumed. On receipt of this command the target device SHALL process the request and (if possible) add the Group ID and Group Name to its Group Table, associate a multicast IP address to the group according to the method specified in Section 2.5.5, and send the response only for requests received via unicast IP addressing. If Group Name is not supported, the Group Name field SHALL be ignored.

Only in the case the request is sent via unicast IPv6 addressing, the target device SHALL respond with a map of response parameters with the key/value pair shown in Table 61.

3398

Name	Key	Type	Encoding
Status	0	8-bit enumeration	Major type 0
Group ID	1	16-bit unsigned integer	Major type 0

3399

**Table 61. Add Group Response Parameters**

3400 The Status field SHALL be set to SUCCESS, FAILURE, DUPLICATE\_EXISTS, or  
 3401 INSUFFICIENT\_SPACE as appropriate. The Group ID field SHALL be set to the Group ID field of the  
 3402 received Add Group command.

3403 If no match exists between the requested group identified by Group ID and the Group ID entries in the  
 3404 Group Table Status SHALL be set to SUCCESS. If a match exists between the requested group and the  
 3405 entries in the Group Table, Status SHALL be set to DUPLICATE\_EXISTS. If the Group Table does not  
 3406 have enough capacity to store the group to be added, Status SHALL be set to INSUFFICIENT\_SPACE. If  
 3407 the Group ID of the request is outside of the allowed range (0x0001-0xfff7) or the device cannot support  
 3408 the IPv6 address or port specified, the field Status SHALL be set to FAILURE.

3409

## E.1.2 View Group

3410 To view group membership of an endpoint of a target device, the sending device SHALL send a POST  
 3411 request to the URI-Path /zcl/e/<eid>/s4/c/1 with the payload set to a map of command parameters with the  
 3412 key/value pairs shown in Table 62.

Name	Key	Type	Encoding
Group ID	0	16-bit unsigned integer	Major type 0

3413

**Table 62. View Group Command Parameters**

3414 Group ID represents the identifier of the group whose membership is requested The target device SHALL  
 3415 process the request and send a response only to requests sent via unicast IP addressing. The response  
 3416 SHALL have the payload set to a map of response parameters with the key/value pairs shown in Table 63.

Name	Key	Type	Encoding
Status	0	8-bit enumeration	Major type 0
Group ID	1	16-bit unsigned integer	Major type 0
Group Name	2	Character String	Major type 3

3417

**Table 63. View Group Response Parameters**

3418 The Status field SHALL be set to SUCCESS or NOT\_FOUND as appropriate. If a match exists between  
 3419 the Group ID in the request and a Group ID in the entries in the Group Table, the Status field SHALL be  
 3420 set to SUCCESS. If a match does not exist between the Group ID in the request and a Group ID the Status  
 3421 SHALL be set to NOT\_FOUND.

3422 The Group ID field SHALL be set to the Group ID field of the received View Group command. If the  
 3423 status is SUCCESS, and group names are supported, the Group Name field is set to the Group Name  
 3424 associated with that Group ID in the Group Table; otherwise it is set to a zero-length string. The Group  
 3425 Name field SHALL be set to an empty string if the Status field is not set to SUCCESS.

3426 The device MAY include the Address Assignment Mode, IPv6 Address and Group Port extensions. If these  
 3427 are not included, the default value specified in the extensions field SHALL be assumed.

### 3428 E.1.3 Get Group Membership

3429 To view group membership of an endpoint of a target device, the sending device SHALL send a POST  
 3430 request to the URI-Path /zcl/e/<eid>/s4/c/2 with the payload set to a map of command parameters with the  
 3431 key/value pairs shown in Table 64.

Name	Key	Type	Encoding
Group List	0	Array of 16-bit integers	Major type 4

3432 **Table 64. Get Group Membership Command Parameters**

3433 If the group list field contains at least one Group ID of which target device is a member, the target device  
 3434 SHALL respond with each Group ID that match a group in the group list field.

3435 If the group list field does not contain any group of which the entity is a member, the entity SHALL only  
 3436 respond if the command is received via unicast IP addressing.

3437 The target device SHALL respond with the payload set to a map of response parameters with the key/value  
 3438 pairs shown in Table 65.

3439

Name	Key	Type	Encoding
Capacity	0	Unsigned 8-bit integer	Major type 0
Group List	1	List of 16-bit group ID	Major type 4

3440 **Table 65. Get Group Membership Response Parameters**

3441 The Capacity field SHALL contain the remaining capacity of the group table of the device. The following  
 3442 values apply:

3443

0	No further groups MAY be added.
0 < Capacity < 0xfe	Capacity holds the number of groups that MAY be added.
0xfe	At least one further group MAY be added (exact number is unknown)
0xff	It is unknown if any further groups MAY be added.

3444

The Group List field SHALL contain the identifiers either of all the groups in the group table (in the case where the Group List field of the received Get Group Membership command was empty) or all the groups from the Group List field of the received Get Group Membership command which are in the group table. If the total number of groups will cause the maximum payload length of a Zigbee packet to be exceeded, then the Group List field SHALL contain only as many groups as will fit.

## E.1.4 Remove Group

To remove a group membership of an endpoint of a target device, the sending device SHALL send a POST request to the URI-Path /zcl/e/<eid>/s4/c/3 with the payload set to a map of command parameters with the key/value pairs shown in Table 66.

Name	Key	Type	Encoding
Group ID	0	16-bit unsigned integer	Major type 0

**Table 66. Remove Group Command Parameters**

Group ID represents the identifier of the group requested to be removed with this command.

On receipt of this command the target device SHALL process the request and (if possible) remove the group identified by the received Group ID from its Group Table, remove the multicast IP address associated with the group, and respond only for requests received via unicast IP addressing.

The response SHALL have the payload set to a map of response parameters with the key/value pairs shown in Table 67.

Name	Key	Type	Encoding
Status	0	8-bit enumeration	Major type 0
Group ID	1	16-bit unsigned integer	Major type 0

**Table 67. Remove Group Response Parameters**

The Status field SHALL be set to SUCCESS or NOT FOUND as appropriate. The Group ID field SHALL be set to the Group ID field of the received Remove Group command.

Only in the case the request is send via unicast IP addressing, the following cases apply. If a match exists between the Group ID in the request and a Group ID in the entries in the Group Table, the target device SHALL remove the entry from the Group Table and respond with 2.04 (Changed) with Status set to SUCCESS. If a match does not exists between the Group ID in the request and a Group ID in the entries in the Group Table, the target device SHALL respond with 2.04 (Changed) with Status set to NOT\_FOUND.

If the Group ID of the request is outside of the allowed range (0x0001-0xffff7), the target device SHALL NOT process the request and respond with 2.04 (Content) with the field Status set to INVALID\_FIELD.

## E.1.5 Remove All Groups

To remove all group memberships to an endpoint of a target device, the sending device SHALL send a POST request with empty payload to the URI-Path /zcl/e/<eid>/s4/c/4.



3474 The target device SHALL process the request and either remove all group membership on the specified  
 3475 endpoint and respond with 2.04 (i.e. Changed) with empty payload or respond with 5.00 (i.e. Internal  
 3476 Server Error) with empty payload. If no groups exist other than the broadcast group, the device SHALL  
 3477 respond with 2.04 (i.e. Changed) with empty payload or 5.00 (i.e. Internal Server Error) with empty  
 3478 payload. In all cases, the broadcast group (Group Id 65535) SHALL NOT be removed.

## 3479 E.1.6 Add Group If Identifying

3480 To add group membership to an endpoint of a target device on condition that it is identifying itself, the  
 3481 sending device SHALL send a POST request to the URI-Path /zcl/e/<eid>/s4/c/5 with the payload set to a  
 3482 map of command parameters with the key/value pairs shown in Table 60 (i.e. same payload and command  
 3483 parameters as for the Add Group command).

3484 The target device SHALL process the request by first checking whether it is currently identifying itself. If  
 3485 so then the device SHALL (if possible) add the Group ID and Group Name to its Group Table. If the device  
 3486 it not currently identifying itself then no action SHALL be taken.

3487 No response is defined as this command is expected to be sent via multicast IP addressing.

## 3488 E.1.7 Command Extensions

3489

Name	Tag Id	Type	Encoding
Address Assignment Mode	1	8-bit unsigned integer	Major type 0
Multicast IPv6 Address	2	8-bit or 128-bit	Major type 2
Group Port	3	16-bit unsigned integer	Major type 0

3490

### 3491 E.1.7.1 Address Assignment Mode

3492 Address Assignment Mode indicates the method that SHALL be used to associate the Group ID of a  
 3493 multicast group with the corresponding multicast IPv6 address as described in Section 2.5.5. The following  
 3494 values SHALL apply:

3495 0x00-0x10 For automatic assignment mode with the *base* parameter indicated by the value.

3496 0xfe For manual assignment mode.

3497 No other values SHALL be allowed. The value of *base* parameter SHOULD be 0x0a, as specified in  
 3498 Section 2.5.5. If the value is omitted, a default base of 0x0a SHALL be applied.

3499 The address assignment mode field MAY be used to extend the Add Group and View Group Response  
 3500 payloads.

### 3501 E.1.7.2 Multicast IPv6 Address

3502 The Multicast IPv6 address parameter contains the necessary information to interpret the IPv6 associated  
 3503 with the group identified by the Group ID. For automatic assignment mode, this parameter corresponds to

3504 an 8-bit string representing the 0 R P T flags and scope field of the IPv6 multicast address as specified in  
3505 Section 2.7 of [[IPV6ADDR](#)].

3506 For automatic assignment mode, this parameter MAY be excluded. In this case, default 0 R P T flags and  
3507 scope as specified in Section 2.5.6 SHALL be used by the target device. For manual assignment mode, this  
3508 parameter corresponds to a 128-bit string representing the full IPv6 multicast address.

3509 For manual assignment mode, this parameter MUST be included.

3510 The multicast IPv6 address field MAY be used to extend the Add Group and View Group Response  
3511 payloads.

### 3512 **E.1.7.3 Group Port**

3513 The Group Port parameter contains the destination port associated with the IPv6 multicast address. Devices  
3514 MAY choose to specify a destination port for a multicast group to avoid collision with possible other  
3515 applications using the same multicast IPv6 addresses in the same network. Alternatively, the Group Port  
3516 parameter MAY be excluded. In this case standard *coap* port (5683) SHALL be used by the target device.

3517 The group port field MAY be used to extend the Add Group and View Group Response payloads.

## 3518 **E.2 Scenes Cluster**

3519 Scene operations are implemented as specified in ZCL with the following clarification of extension field set  
3520 representation.

### 3521 **E.2.1 Extension Field Set**

3522 The extension field set SHALL be represented as a map by devices implementing this specification. For  
3523 each cluster which has attribute values to be extended, an entry SHALL be made in the map with the key  
3524 set to the cluster identifier, and the value to a map. The internal map SHALL have an entry for each  
3525 extension field. The key SHALL be the attribute id of the extension field, and the value SHALL be the  
3526 desired value for operation of a scene.

3527 Currently this applies to Add Scene, View Scene Response, and Enhanced View Scene Response messages.

3528

## Annex F For Future Consideration

3529

### F.1 Operation Over IPv4 Networks

3530

3531

The initial specification of ZCLIP focuses on operation over IPv6 networks. Additional consideration is needed to specify how ZCLIP will operate over IPv4 networks.

3532