



OCF 2.3 – Z-Wave Data Model Mapping specification – BTG

Legal Disclaimer

THIS IS A DRAFT SPECIFICATION DOCUMENT ONLY AND HAS NOT BEEN ADOPTED BY THE OPEN CONNECTIVITY FOUNDATION. THIS DRAFT DOCUMENT MAY NOT BE RELIED UPON FOR ANY PURPOSE OTHER THAN REVIEW OF THE CURRENT STATE OF THE DEVELOPMENT OF THIS DRAFT DOCUMENT. THE OPEN CONNECTIVITY FOUNDATION AND ITS MEMBERS RESERVE THE RIGHT WITHOUT NOTICE TO YOU TO CHANGE ANY OR ALL PORTIONS HEREOF, DELETE PORTIONS HEREOF, MAKE ADDITIONS HERETO, DISCARD THIS DRAFT DOCUMENT IN ITS ENTIRETY OR OTHERWISE MODIFY THIS DRAFT DOCUMENT AT ANY TIME. YOU SHOULD NOT AND MAY NOT RELY UPON THIS DRAFT DOCUMENT IN ANY WAY, INCLUDING BUT NOT LIMITED TO THE DEVELOPMENT OF ANY PRODUCTS OR SERVICES. IMPLEMENTATION OF THIS DRAFT DOCUMENT IS DONE AT YOUR OWN RISK AMEND AND IT IS NOT SUBJECT TO ANY LICENSING GRANTS OR COMMITMENTS UNDER THE OPEN CONNECTIVITY FOUNDATION INTELLECTUAL PROPERTY RIGHTS POLICY OR OTHERWISE. IN CONSIDERATION OF THE OPEN CONNECTIVITY FOUNDATION GRANTING YOU ACCESS TO THIS DRAFT DOCUMENT, YOU DO HEREBY WAIVE ANY AND ALL CLAIMS ASSOCIATED HERewith INCLUDING BUT NOT LIMITED TO THOSE CLAIMS DISCUSSED BELOW, AS WELL AS CLAIMS OF DETRIMENTAL RELIANCE.

The OCF logo is a trademark of Open Connectivity Foundation, Inc. in the United States or other countries. *Other names and brands may be claimed as the property of others.

Copyright © 2018 Open Connectivity Foundation, Inc. All rights reserved.

Copying or other form of reproduction and/or distribution of these works are strictly prohibited.



27

28

CONTENTS

29	1	Scope.....	6
30	2	Normative references	6
31	3	Terms, definitions and symbols	7
32	3.1	Terms and definitions.....	7
33	3.2	Terms and definitions.....	7
34	3.3	Symbols and abbreviations	7
35	4	Document conventions and organization.....	7
36	4.1	Introduction.....	7
37	4.2	Conventions.....	7
38	4.3	Notation.....	7
39	4.4	Data types	8
40	5	Theory of Operation.....	8
41	5.1	Interworking Approach	8
42	5.2	Mapping Syntax	8
43	5.2.1	General	8
44	5.2.2	Value Assignment.....	8
45	5.2.3	Property Naming.....	8
46	5.2.4	Arrays.....	8
47	5.2.5	Default Mapping	9
48	5.2.6	Conditional Mapping	9
49	5.2.7	Loops	9
50	5.2.8	Command Invocation	9
51	6	Device Type Mapping	9
52	6.1	Introduction.....	9
53	6.2	Z-Wave Device Types to OCF Device Types.....	9
54	7	Resource to Command Class Mapping	10
55	7.1	Introduction.....	10
56	7.2	Z-Wave Command Classes to OCF Resources	10
57	7.2.1	Battery Command Class Mapping	11
58	7.2.2	Binary Switch Command Class Mapping.....	11
59	7.2.3	Door Lock Command Class Mapping	12
60	7.2.4	Multilevel Sensor Command Class Mapping	12
61	7.2.5	Multilevel Switch Command Class Mapping	12
62	7.2.6	Notification Command Class Mapping.....	12
63	7.2.7	User Code Command Class Mapping.....	13
64	8	Detailed Mapping APIs	13
65	8.1	Battery Command Class	14
66	8.1.1	Derived model	14
67	8.1.2	Property definition	14
68	8.1.3	Derived model definition	14



69	8.2	Binary Switch Command Class	15
70	8.2.1	Derived model	15
71	8.2.2	Property definition	15
72	8.2.3	Derived model definition	15
73	8.3	Door Lock Command Class.....	16
74	8.3.1	Derived model	16
75	8.3.2	Property definition	16
76	8.3.3	Derived model definition	16
77	8.4	Multilevel Sensor Command Class Carbon Dioxide	17
78	8.4.1	Derived model	17
79	8.4.2	Property definition	17
80	8.4.3	Derived model definition	17
81	8.5	Multilevel Sensor Command Class Carbon Monoxide.....	19
82	8.5.1	Derived model	19
83	8.5.2	Property definition	19
84	8.5.3	Derived model definition	19
85	8.6	Multilevel Sensor Command Class Smoke Density.....	21
86	8.6.1	Derived model	21
87	8.6.2	Property definition	21
88	8.6.3	Derived model definition	21
89	8.7	Multilevel Sensor Command Class Water Flow	23
90	8.7.1	Derived model	23
91	8.7.2	Property definition	23
92	8.7.3	Derived model definition	24
93	8.8	Multilevel Switch Command Class.....	25
94	8.8.1	Derived model	25
95	8.8.2	Property definition	25
96	8.8.3	Derived model definition	25
97	8.9	Notification Command Class	26
98	8.9.1	Derived model	26
99	8.9.2	Property definition	26
100	8.9.3	Derived model definition	27
101	8.10	User Code Command Class	30
102	8.10.1	Derived model	30
103	8.10.2	Property definition	30
104	8.10.3	Derived model definition	30
105			
106			



107

Figures

108

109

No table of figures entries found.



110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140

Tables

Table 1. Z-Wave to OCF Device Type Mapping.....	9
Table 2. Z-Wave Command Class to OCF Resource Type Mapping	10
Table 3. Command Class to Resource Summary	13
Table 4 The property mapping for zwave.operation.batterycommandclass	14
Table 5 The properties of zwave.operation.batterycommandclass	14
Table 6 The property mapping for zwave.operation.binaryswitchcommandclass	15
Table 7 The properties of zwave.operation.binaryswitchcommandclass	15
Table 8 The property mapping for zwave.operation.doorlockcommandclass	16
Table 9 The properties of zwave.operation.doorlockcommandclass	16
Table 10 The property mapping for zwave.operation.multilevelsensorcommandclasscarbondioxide	17
Table 11 The properties of zwave.operation.multilevelsensorcommandclasscarbondioxide ...	17
Table 12 The property mapping for zwave.operation.multilevelsensorcommandclasscarbonmonoxide	19
Table 13 The properties of zwave.operation.multilevelsensorcommandclasscarbonmonoxide	19
Table 14 The property mapping for zwave.operation.multilevelsensorcommandclasssmokedensity	21
Table 15 The properties of zwave.operation.multilevelsensorcommandclasssmokedensity ...	21
Table 16 The property mapping for zwave.operation.multilevelsensorcommandclasswaterflow	23
Table 17 The properties of zwave.operation.multilevelsensorcommandclasswaterflow	23
Table 18 The property mapping for zwave.operation.multilevelswitchcommandclass	25
Table 19 The properties of zwave.operation.multilevelswitchcommandclass	25
Table 20 The property mapping for zwave.operation.notificationcommandclass	26
Table 21 The properties of zwave.operation.notificationcommandclass	27
Table 22 The property mapping for zwave.operation.usercodecommandclass	30
Table 23 The properties of zwave.operation.usercodecommandclass	30



141 **1 Scope**

142 The OCF Z-Wave Data Model specification (“this specification”) provides detailed mapping
143 information to provide equivalency between Z-Wave Command Classes and OCF defined
144 Resources.

145 This specification provides mapping for Device Types (Z-Wave to OCF), identifies OCF Resources
146 for both Z-Wave Command Classes and for each Command Class defines the detailed Property by
147 Property mapping using OCF defined extensions to JSON schema to programmatically define the
148 mappings.

149 **2 Normative references**

150 The following documents, in whole or in part, are normatively referenced in this document and are
151 indispensable for its application. For dated references, only the edition cited applies. For undated
152 references, the latest edition of the referenced document (including any amendments) applies.

153 JSON Schema Core, *JSON Schema: core definitions and terminology*, January 2013
154 <http://json-schema.org/latest/json-schema-core.html>

155 JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January 2013
156 <http://json-schema.org/latest/json-schema-validation.html>

157 JSON Hyper-Schema, *JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON*,
158 October 2016
159 <http://json-schema.org/latest/json-schema-hypermedia.html>

160 OCF Core Specification, *Open Connectivity Foundation Core Specification, Version 2.0*
161 https://openconnectivity.org/specs/OCF_Core_Specification_v2.0.0.pdf

162 OCF Device Specification, *Open Connectivity Foundation Device Specification, Version 2.0*
163 https://openconnectivity.org/specs/OCF_Device_Specification_v2.0.0.pdf

164 OCF Resource Type Specification, *Open Connectivity Foundation Resource Type Specification,*
165 *Version 2.0*
166 https://openconnectivity.org/specs/OCF_Resource_Type_Specification_v2.0.0.pdf

167 OCF Security Specification, *Open Connectivity Foundation Security Specification, Version 2.0*
168 https://openconnectivity.org/specs/OCF_Security_Specification_v2.0.0.pdf

169 OCF Bridging Specification, *Open Connectivity Foundation Bridging Specification, Version 2.0*
170 https://openconnectivity.org/specs/OCF_Bridging_Specification_v1.3.0.pdf

171 RAML Specification, *RESTful API Modeling Language, Version 0.8*
172 <https://github.com/raml-org/raml-spec/blob/master/versions/raml-08/raml-08.md>

173 OpenAPI Specification, Version 2.0
174 <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md>

175 Derived Models for Interoperability between IoT Ecosystems, Stevens & Merriam, March 2016
176 [https://www.iab.org/wp-content/IAB-uploads/2016/03/OCF-Derived-Models-for-Interoperability-](https://www.iab.org/wp-content/IAB-uploads/2016/03/OCF-Derived-Models-for-Interoperability-Between-IoT-Ecosystems_v2-examples.pdf)
177 [Between-IoT-Ecosystems_v2-examples.pdf](https://www.iab.org/wp-content/IAB-uploads/2016/03/OCF-Derived-Models-for-Interoperability-Between-IoT-Ecosystems_v2-examples.pdf)

178 Z-Wave Plus Device and Command Class Types and Defines Specification

179 [https://www.silabs.com/documents/login/miscellaneous/SDS11847-Z-Wave-Plus-Device-Type-](https://www.silabs.com/documents/login/miscellaneous/SDS11847-Z-Wave-Plus-Device-Type-Specification.pdf)
180 [Specification.pdf](https://www.silabs.com/documents/login/miscellaneous/SDS11847-Z-Wave-Plus-Device-Type-Specification.pdf)

181



182 Z-Wave Plus v2 Device Type Specification

183 [https://www.silabs.com/documents/login/miscellaneous/SDS14224-Z-Wave-Plus-v2-Device-Type-](https://www.silabs.com/documents/login/miscellaneous/SDS14224-Z-Wave-Plus-v2-Device-Type-Specification.pdf)
184 [Specification.pdf](https://www.silabs.com/documents/login/miscellaneous/SDS14224-Z-Wave-Plus-v2-Device-Type-Specification.pdf)

185 **3 Terms, definitions and symbols**

186 All terms and definitions as defined in the OCF Core Specification, OCF Security Specification, and
187 OCF Security Specification OCF Bridging Specification also apply to this specification.

188 **3.1 Terms and definitions**

189 As defined in the OCF Core Specification and OCF Bridging Specification with the following
190 additions.

191 **3.2 Terms and definitions**

192 None defined.

193 **3.3 Symbols and abbreviations**

194 None defined.

195 **4 Document conventions and organization**

196 **4.1 Introduction**

197 For the purposes of this document, the terms and definitions given in OCF Core Specification and
198 OCF Security Specification apply.

199 **4.2 Conventions**

200 In this specification a number of terms, conditions, mechanisms, sequences, parameters, events,
201 states, or similar terms are printed with the first letter of each word in uppercase and the rest
202 lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal
203 technical English meaning

204 **4.3 Notation**

205 In this document, features are described as required, recommended, allowed or DEPRECATED as
206 follows:

207 Required (or shall or mandatory).

208 – These basic features shall be implemented to comply with OIC Core Architecture. The phrases
209 “shall not”, and “PROHIBITED” indicate behaviour that is prohibited, i.e. that if performed means
210 the implementation is not in compliance.

211 Recommended (or should).

212 – These features add functionality supported by OIC Core Architecture and should be
213 implemented. Recommended features take advantage of the capabilities OIC Core Architecture,
214 usually without imposing major increase of complexity. Notice that for compliance testing, if a
215 recommended feature is implemented, it shall meet the specified requirements to be in
216 compliance with these guidelines. Some recommended features could become requirements in
217 the future. The phrase “should not” indicates behaviour that is permitted but not recommended.

218 Allowed (or allowed).

219 – These features are neither required nor recommended by OIC Core Architecture, but if the
220 feature is implemented, it shall meet the specified requirements to be in compliance with these
221 guidelines.



222 – Conditionally allowed (CA) The definition or behaviour depends on a condition. If the specified
223 condition is met, then the definition or behaviour is allowed, otherwise it is not allowed.

224 Conditionally required (CR)

225 – The definition or behaviour depends on a condition. If the specified condition is met, then the
226 definition or behaviour is required. Otherwise the definition or behaviour is allowed as default
227 unless specifically defined as not allowed.

228 DEPRECATED

229 – Although these features are still described in this specification, they should not be implemented
230 except for backward compatibility. The occurrence of a deprecated feature during operation of
231 an implementation compliant with the current specification has no effect on the
232 implementation's operation and does not produce any error conditions. Backward compatibility
233 may require that a feature is implemented and functions as specified but it shall never be used
234 by implementations compliant with this specification.

235 Strings that are to be taken literally are enclosed in “double quotes”.

236 Words that are emphasized are printed in *italic*.

237 **4.4 Data types**

238 Data types are defined in the OCF Core Specification.

239 **5 Theory of Operation**

240 **5.1 Interworking Approach**

241 The interworking between Z-Wave defined Command Classes and OCF defined Resource Types
242 is modelled using the derived model syntax described in Derived Models for Interoperability.

243 **5.2 Mapping Syntax**

244 Within the defined syntax for derived modelling used by this Specification there are two blocks that
245 define the actual Property-Property equivalence or mapping. These blocks are identified by the
246 keywords ‘x-to-ocf’ and ‘x-from-ocf’. Derived Models for Interoperability does not define a rigid
247 syntax for these blocks; they are free form string arrays that contain pseudo-coded mapping logic.
248 Within this specification we apply the rules in the following sub-sections to these blocks to ensure
249 consistency and re-usability and extensibility of the mapping logic that is defined.

250 **5.2.1 General**

251 All statements are terminated with a carriage return.

252 **5.2.2 Value Assignment**

253 The equals sign (=) is used to assign one value to another. The assignee is on the left of the
254 operator; the value being assigned on the right.

255 **5.2.3 Property Naming**

256 All Property names are identical to the name used by the original model; for example, from the
257 OCF Temperature Resource the Property name ‘temperature’ is used whereas when referred to
258 the derived ecosystem then the semantically equivalent Property name is used.

259 When the same name is used by both OCF and the derived ecosystem for semantically equivalent
260 values then the name of the OCF defined Property is prepended by the ecosystem designator ‘ocf’
261 to avoid ambiguity (e.g. ‘ocf.step’)

262 **5.2.4 Arrays**

263 An array element is indicated by the use of square brackets ‘[]’ with the index of the element
264 contained therein, e.g. range[1]. All arrays start at an index of 0. If an entire array is being
265 referenced then no index is included, e.g. selectablehumiditylevels[].



266 **5.2.5 Default Mapping**

267 There are cases where the specified mapping is not possible as one or more of the Properties
268 being mapped is optional in the source model. In all such instances a default mapping is provided.
269 The default map is indicated by the prepending of an 'otherwise:' modifier to the assignment. (e.g.
270 'otherwise: step = 1')

271 **5.2.6 Conditional Mapping**

272 When a mapping is dependent on the meeting of other conditions then the syntax:

273 if 'condition', 'mapping'.

274 Is applied.

275 E.g. if step >0, ocf.step = step.

276 **5.2.7 Loops**

277 When a mapping can be represented by a repeated loop governed by some condition then the
278 syntax:

279 for 'initialize', 'condition', 'increment': 'mapping'

280 Where:

281 'initialize' is an initial local loop control variable setting.

282 'condition' is the loop controller, the loop repeats until the condition evaluates to 'false'.

283 'increment' allows for update of the control variable, if omitted an increment of '1' is assumed.

284 Is applied.

285 E.g. for x=0, x < sizeof(supportedmodes): ocf.supportedmodes[x] =
286 modearray[supportedmodes[x]]

287 **5.2.8 Command Invocation**

288 The invocation of a command from the derived ecosystem as part of the mapping from an OCF
289 Resource is indicated by the use if a double colon '::' delimiter between the applicable Command
290 Class or other construct identifier and the command name. The command name always includes
291 trailing parentheses which would include any parameters should they be passed.

292 **6 Device Type Mapping**

293 **6.1 Introduction**

294 This Section contains the mappings to Device Types.

295 **6.2 Z-Wave Device Types to OCF Device Types**

296 The following table captures the mapping between Z-Wave Plus defined Device Types (see Z-Wave
297 Plus Device Type Specification) and OCF defined Device Types (see OCF Device Specification).

298 **Table 1. Z-Wave to OCF Device Type Mapping**

Classification of Z-Wave Generic Type	Z-Wave Device Type	ZWave Device Type ID	OCF Device Type
---------------------------------------	--------------------	----------------------	-----------------



Multilevel Switch	Light Dimmer Switch	0x01	oic.d.light
Entry Control	DoorLock - Keypad	0x03	oic.d.smartlock
Binary Switch	On/Off Power Switch	0x01	oic.d.switch
Multilevel Sensor	Sensor - Multilevel	0x01	oic.d.sensor
Notification Sensor	Sensor - Notification	0x01	oic.d.sensor

299 Z-Wave Plus v2 device types are equivalently mapped to the Z-Wave Plus device types as specified
 300 in Z-Wave Plus v2 Device Type Specification.

301 7 Resource to Command Class Mapping

302 7.1 Introduction

303 Clause 7 lists the set of applicable Z-Wave Command Classes and provides the OCF Resource
 304 Type(s) to which the Command Classes map along an introduction the semantics of the mapping.
 305 The detailed mappings are provided in clause 8.

306 7.2 Z-Wave Command Classes to OCF Resources

307 The following tables capture the mapping between Z-Wave Command Classes and OCF defined
 308 Resource Types. Detailed Property by Property mappings are provided in Section 8.

309 Table 2 captures the mappings for Command Classes for a Z-Wave Device.

310

311

Table 2. Z-Wave Command Class to OCF Resource Type Mapping

Z- Wave Plus Device	Z-Wave Command Class	OCF Resource Type	OCF Device Type	OCF Device Name
Light Dimmer Switch	Multilevel Switch Command Class	oic.r.switch.binary	oic.d.light	Light
	Multilevel Switch Command Class	oic.r.light.dimming		
	Manufacturer Specific Command Class	oic.wk.d		
	Version Command Class Z-Wave Plus Info Command Class	oic.wk.p		
Door Lock – Keypad	Door Lock Command Class	oic.r.lock.status	oic.d.smartlock	Smart Lock
	User Code Command Class	oic.r.lock.code		
	Battery Command Class	oic.r.energy.battery.		
	Manufacturer Specific Command Class	oic.wk.d		
	Version Command Class Z-Wave Plus Info Command Class	oic.wk.p		



On/Off Power Switch	Binary Switch Command Class	oic.r.switch.binary	oic.d.switch	Switch
	Battery Command Class	oic.r.energy.battery.		
	Manufacturer Specific Command Class	oic.wk.d		
	Version Command Class Z-Wave Plus Info Command Class	oic.wk.p		
Sensor - Multilevel	Multilevel Sensor Command Class	oic.r.sensor.carbondioxide	oic.d.sensor	Generic Sensor
	Multilevel Sensor Command Class	oic.r.sensor.carbonmonoxide		
	Multilevel Sensor Command Class	oic.r.sensor.water		
	Multilevel Sensor Command Class	oic.r.sensor.smoke		
	Battery Command Class	oic.r.energy.battery.		
	Manufacturer Specific Command Class Version Command Class Z-Wave Plus Info Command Class	oic.wk.d oic.wk.p		
Sensor - Notification	Notification Command Class	oic.r.sensor.carbondioxide	oic.d.sensor	Generic Sensor
	Notification Command Class	oic.r.sensor.carbonmonoxide		
	Notification Command Class	oic.r.sensor.water		
	Notification Command Class	oic.r.sensor.smoke		
	Battery Command Class	oic.r.energy.battery.		
	Manufacturer Specific Command Class Version Command Class Z-Wave Plus Info Command Class	oic.wk.d oic.wk.p		

312

313 7.2.1 Battery Command Class Mapping

314 This API defines the mapping between an instance of a Battery Command Class and the OCF
315 Battery Energy Resource. Note that the setting of the Value of OCF

316 Battery Energy to 'charge' is handled via the 'Battery Level' of Battery Command Class. A
317 RETRIEVE on a Battery Energy maps to Battery Get Command on an instance of a Z-Wave Battery
318 Command Class.

319

320 7.2.2 Binary Switch Command Class Mapping

321 This API defines the mapping between an instance of a Z-Wave Binary Switch Command Class
322 and an OCF Binary Switch Resource. Note that the setting of the Value of OCF Binary Switch to
323 '0x00' (off) and '0x255(on)' is handled via the 'Value' of Binary Switch Command Class. A
324 RETRIEVE on a Binary Switch maps to Binary Switch Get Command on an instance of a Z-Wave



325 Binary Switch Command Class. And a UPDATE on a Binary Switch maps to Binary Switch Set
326 Command on an instance of a Z-Wave Binary Switch Command Class.

327 **7.2.3 Door Lock Command Class Mapping**

328 This API defines the mapping between an instance of a Door Lock Command Class and the OCF
329 Door Resource. Note that the setting of the Value of OCF Lock Status is handled via the 'Value'
330 "Door Unsecured"(0x00) and "Door Secured"(0xFF) of Door Lock Command Class. A RETRIEVE
331 on a Door maps to Door Lock Operation Get Command on an instance of a Z-Wave Door Lock
332 Command Class. And a UPDATE on a Door maps to Door Lock Operation Set Command on an
333 instance of a Z-Wave Door Lock Command Class.

334 **7.2.4 Multilevel Sensor Command Class Mapping**

335 **7.2.4.1 Multilevel Sensor Command Class Mapping for Carbon Dioxide Sensor**

336 This API defines the mapping between an instance of a Z-Wave Multilevel Sensor Command Class
337 and an OCF Carbon Dioxide sensor resource. Multilevel Sensor Command Class has 5 properties:
338 Sensor Type, Precision, Scale, Size, and Sensor Value. In case Sensor Type is a carbon dioxide
339 sensor, an OCF Carbon Dioxide sensor resource is mapped. A RETRIEVE on a Carbon Dioxide
340 sensor maps to Multilevel Sensor Get Command on an instance of a Z-Wave Multilevel Sensor
341 Command Class.

342 **7.2.4.2 Multilevel Sensor Command Class Mapping for Carbon Monoxide Sensor**

343 This API defines the mapping between an instance of a Z-Wave Multilevel Sensor Command Class
344 and an OCF Carbon Monoxide sensor resource. Multilevel Sensor Command Class has 5 properties:
345 Sensor Type, Precision, Scale, Size, and Sensor Value. In case Sensor Type is a carbon monoxide
346 sensor, an OCF Carbon Monoxide sensor resource is mapped. A RETRIEVE on a Carbon Monoxide
347 sensor maps to Multilevel Sensor Get Command on an instance of a Z-Wave Multilevel Sensor
348 Command Class.

349 **7.2.4.3 Multilevel Sensor Command Class Mapping for Smoke Density Sensor**

350 This API defines the mapping between an instance of a Z-Wave Multilevel Sensor Command Class
351 and an OCF Smoke sensor resource. Multilevel Sensor Command Class has 5 properties: Sensor
352 Type, Precision, Scale, Size, and Sensor Value. In case Sensor Type is a smoke density sensor,
353 an OCF Smoke sensor resource is mapped. A RETRIEVE on a Smoke sensor maps to Multilevel
354 Sensor Get Command on an instance of a Z-Wave Multilevel Sensor Command Class.

355 **7.2.4.4 Multilevel Sensor Command Class Mapping for Water Flow Sensor**

356 This API defines the mapping between an instance of a Z-Wave Multilevel Sensor Command Class
357 and an OCF Water sensor resource. Multilevel Sensor Command Class has 5 properties: Sensor
358 Type, Precision, Scale, Size, and Sensor Value. In case Sensor Type is a water flow sensor, an
359 OCF Water sensor resource is mapped. A RETRIEVE on a water sensor maps to Multilevel Sensor
360 Get Command on an instance of a Z-Wave Multilevel Sensor Command Class.

361 **7.2.5 Multilevel Switch Command Class Mapping**

362 This API defines the mapping between an instance of a Z-Wave Multilevel Switch Command Class
363 and an OCF Binary Switch Resource or an OCF Dimming Light Resource depending on the 'Value'
364 of Multilevel Switch Set Command of Multilevel Switch Command Class. Note that the setting of
365 the Value of OCF Binary Switch to '0x00' (off) and '0x63' (on) and the Value of OCF Dimming Light
366 to 1 (min) and 99 (max) is handled via the 'Value' of Multilevel Switch Set. A RETRIEVE on a Binary
367 Switch or Dimming Light maps to Multilevel Switch Get Command on an instance of a Z-Wave
368 Multilevel Switch Command Class. And a UPDATE on a Binary Switch or Dimming Light maps to
369 Multilevel Switch Set Command on an instance of a Z-Wave Multilevel Switch Command Class.

370 **7.2.6 Notification Command Class Mapping**

371 This API defines the mapping between an instance of a Notification Command Class and OCF
372 Specific sensor resources. Notification Command Class has 9 properties; these map as follows: V1
373 Alarm Type, V1 Alarm Level, Notification Status, Notification Type, Notification Event:State,



374 Sequence, Event:State Parameters Length, Event:State Parameter, Sequence Number =>
 375 corresponding properties of smoke sensor, carbon monoxide sensor, carbon dioxide sensor or
 376 water sensor. This is presented in OCF as the distinct Resource instances. A RETRIEVE on a
 377 Specific Sensor maps to Notification Get Command on an instance of a Z-Wave Notification
 378 Command Class. And a UPDATE on a Specific Sensor maps to Notification Set Command on an
 379 instance of a Z-Wave Notification Command Class.

380 7.2.7 User Code Command Class Mapping

381 This API defines the mapping between an instance of a User Code Command Class and the OCF
 382 Lock Code Resource. A RETRIEVE on a Lock Code maps to User Code Get Command on an
 383 instance of a Z-Wave User Code Command Class. And a UPDATE on a Lock Code maps to User
 384 Code Set Command on an instance of a Z-Wave User Code Command Class.

385 8 Detailed Mapping APIs

386 This section provides a mapping description (using JSON that aligns with the Derived Modelling
 387 syntax described in [Derived Model White Paper]) for all Command Classes and Resources that
 388 are within scope.

389 Table 3 provides a reference and link to the per Command Class sub-sections.

390

391

Table 3. Command Class to Resource Summary

Z-Wave Command Class Name	Mapped Resource(s)	Mapping Section
Battery Command Class	oic.r.energy.battery	8.1
Binary Switch Command Class	oic.r.switch.binary	8.2
Door Lock Command Class	oic.r.lock.status	8.3
Multilevel Sensor Command Class	oic.r.sensor.carbondioxide	8.4
Multilevel Sensor Command Class	oic.r.sensor.carbonmonoxide	8.5
Multilevel Sensor Command Class	oic.r.sensor.water	8.6
Multilevel Sensor Command Class	oic.r.sensor.smoke	8.7
Multilevel Switch Command Class	oic.r.switch.binary	8.8
	oic.r.light.dimming	
Notification Command Class	oic.r.sensor.carbondioxide	8.9
Notification Command Class	oic.r.sensor.carbonmonoxide	
Notification Command Class	oic.r.sensor.water	



Notification Command Class	oic.r.sensor.smoke	
User Code Command Class	oic.r.lock.status	8.10

392

393 8.1 Battery Command Class

394 8.1.1 Derived model

395 The derived model: zwave.operation.batterycommandclass

396 8.1.2 Property definition

397 **Table 4 The property mapping for zwave.operation.batterycommandclass**

ZWave Property name	OCF Resource	To OCF	From OCF
Battery Level	oic.r.energy.battery	if Battery Level = 255, ocf.r.energy.battery.lowbattery = true; ocf.r.energy.battery.charge = 0 .if Battery Level != 255, ocf.r.energy.battery.charge = Battery Level.	N/A

398

Table 5 The properties of zwave.operation.batterycommandclass

ZWave Property name	Type	Required	Description
Battery Level	if Battery Level = 255, string if Battery Level != 255, integer	yes	percentage indicating the battery level or low battery warning

399 8.1.3 Derived model definition

```

400 {
401   "id":
402   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.batterycommandclass.json#",
403   "$schema": "http://json-schema.org/draft-04/schema#",
404   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
405   "title": "Battery Command Class",
406   "definitions": {
407     "zwave.operation.batterycommandclass": {
408       "type": "object",
409       "properties": {
410         "Battery Level": {
411           "type": [
412             "if Battery Level = 255, string",
413             "if Battery Level != 255, integer"
414           ],
415           "description": "percentage indicating the battery level or low battery warning",
416           "x-ocf-conversion": {
417             "x-ocf-alias": "oic.r.energy.battery",
418             "x-to-ocf": [
419               "if Battery Level = 255, ocf.r.energy.battery.lowbattery = true;
420 ocf.r.energy.battery.charge = 0.",
421               "if Battery Level != 255, ocf.r.energy.battery.charge = Battery Level."
422             ],
423             "x-from-ocf": [
424               "N/A"
425             ]
426           }
427         }
428       }

```



```

429     }
430   },
431   "type": "object",
432   "allOf": [
433     {"$ref": "#/definitions/zwave.operation.batterycommandclass"}
434   ],
435   "required": ["Battery Level"]
436 }

```

437 8.2 Binary Switch Command Class

438 8.2.1 Derived model

439 The derived model: zwave.operation.binaryswitchcommandclass

440 8.2.2 Property definition

441 **Table 6 The property mapping for zwave.operation.binaryswitchcommandclass**

ZWave name	Property	OCF Resource	To OCF	From OCF
Value		oic.r.switch.binary	if Value = 255, ocf.r.switch.binary.value = true. if Value != 255, ocf.r.switch.binary.value = false.	if ocf.r.switch.binary.value = false, Value = 0 if ocf.r.switch.binary.value = true, Value = 255

442 **Table 7 The properties of zwave.operation.binaryswitchcommandclass**

ZWave name	Property	Type	Required	Description
Value		boolean	yes	On/Off state at the receiving node

443 8.2.3 Derived model definition

```

444 {
445   "id":
446   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.binaryswitchcommandclass.json#",
447   "$schema": "http://json-schema.org/draft-04/schema#",
448   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
449   "title": "Binary Switch Command Classs",
450   "definitions": {
451     "zwave.operation.binaryswitchcommandclass": {
452       "type": "object",
453       "properties": {
454         "Value": {
455           "type": "boolean",
456           "description": "On/Off state at the receiving node",
457           "x-ocf-conversion": {
458             "x-ocf-alias": "oic.r.switch.binary",
459             "x-to-ocf": [
460               "if Value = 255, ocf.r.switch.binary.value = true.",
461               "if Value != 255, ocf.r.switch.binary.value = false."
462             ],
463             "x-from-ocf": [
464               "if ocf.r.switch.binary.value = false, Value = 0",
465               "if ocf.r.switch.binary.value = true, Value = 255"
466             ]
467           }
468         }
469       }
470     }
471   },
472   "type": "object",
473   "allOf": [
474     {"$ref": "#/definitions/zwave.operation.binaryswitchcommandclass"}
475   ],

```



```
477     "required": ["Value"]
478 }
```

479 8.3 Door Lock Command Class

480 8.3.1 Derived model

481 The derived model: zwave.operation.doorlockcommandclass

482 8.3.2 Property definition

483 **Table 8 The property mapping for zwave.operation.doorlockcommandclass**

ZWave Property name	OCF Resource	To OCF	From OCF
Door Lock Mode	oic.r.lock.status	if Door Lock Mode = 0x00, ocf.r.lock.status.lockState = UnLocked if Door Lock Mode = 0xFF, ocf.r.lock.status.lockState = Locked	if ocf.r.lock.status.lockState = Unlocked, Door Lock Mode = 0x00 if ocf.r.lock.status.lockState = Locked, Door Lock Mode = 0xFF

484 **Table 9 The properties of zwave.operation.doorlockcommandclass**

ZWave Property name	Type	Required	Description
Door Lock Mode	integer	yes	operation mode of the door lock device

485 8.3.3 Derived model definition

```
486 {
487   "id":
488   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.doorlockcommandclass.json#",
489   "$schema": "http://json-schema.org/draft-04/schema#",
490   "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
491   "title": "Door Lock Command Class",
492   "definitions": {
493     "zwave.operation.doorlockcommandclass": {
494       "type": "object",
495       "properties": {
496         "Door Lock Mode": {
497           "type" : "integer",
498           "description": "operation mode of the door lock device",
499           "x-ocf-conversion": {
500             "x-ocf-alias": "oic.r.lock.status",
501             "x-to-ocf": [
502               "if Door Lock Mode = 0x00, ocf.r.lock.status.lockState = UnLocked",
503               "if Door Lock Mode = 0xFF, ocf.r.lock.status.lockState = Locked"
504             ],
505             "x-from-ocf": [
506               "if ocf.r.lock.status.lockState = Unlocked, Door Lock Mode = 0x00",
507               "if ocf.r.lock.status.lockState = Locked, Door Lock Mode = 0xFF"
508             ]
509           }
510         }
511       }
512     },
513   },
514   "type": "object",
515   "allOf": [
516     {"$ref": "#/definitions/zwave.operation.doorlockcommandclass"}
517   ],
518   "required": ["Door Lock Mode"]
519 }
```




520 **8.4 Multilevel Sensor Command Class Carbon Dioxide**

521 **8.4.1 Derived model**

522 The derived model: zwave.operation.multilevelsensorcommandclasscarbondioxide

523 **8.4.2 Property definition**

524 **Table 10 The property mapping for**
525 **zwave.operation.multilevelsensorcommandclasscarbondioxide**

ZWave Property name	OCF Resource	To OCF	From OCF
Precision	oic.r.sensor.carbondioxide	ocf.r.sensor.carbondioxide.precision = Precision	N/A
Sensor Type	oic.r.sensor.carbondioxide	if Sensor Type = Carbon dioxide CO2-level, ocf.rt = oic.r.sensor.carbondioxide.	N/A
Size	oic.r.sensor.carbondioxide	N/A	N/A
Sensor Value	oic.r.sensor.carbondioxide	ocf.r.sensor.carbondioxide.value = trueocf.r.sensor.carbondioxide.measurement = Sensor Value	N/A
Scale	oic.r.sensor.carbondioxide	N/A	Scale = ppm (0x00)

526 **Table 11 The properties of zwave.operation.multilevelsensorcommandclasscarbondioxide**

ZWave Property name	Type	Required	Description
Precision	Number	yes	indicate how many decimal places are included the Sensor Value field
Sensor Type	Integer	yes	specify the carbon dioxide sensor type of the actual sensor reading
Size	enum	yes	indicate the length in bytes of the Sensor Value field
Sensor Value	array	yes	specify the value of the actual sensor reading
Scale	Integer	yes	indicate what scale is used for the actual sensor reading

527 **8.4.3 Derived model definition**

```
528 {
529   "id":
530   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.multilevelsensorcommandclasscarbon
531   dioxide.json#",
532   "$schema": "http://json-schema.org/draft-04/schema#",
533   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
534   "title": "Multilevel Sensor Command Class Carbon Dioxide",
535   "definitions": {
536     "zwave.operation.multilevelsensorcommandclasscarbondioxide": {
537       "type": "object",
538       "properties": {
539         "Sensor Type": {
```



```
540     "type" : "Integer",
541     "description": " specify the carbon dioxide sensor type of the actual sensor reading ",
542     "x-ocf-conversion": {
543       "x-ocf-alias": "oic.r.sensor.carbondioxide",
544       "x-to-ocf": [
545         "if Sensor Type = Carbon dioxide CO2-level, ocf.rt = oic.r.sensor.carbondioxide."
546       ],
547       "x-from-ocf": [
548         "N/A"
549       ]
550     },
551   },
552   "Precision": {
553     "type" : "Number",
554     "description": " indicate how many decimal places are included the Sensor Value field ",
555     "x-ocf-conversion": {
556       "x-ocf-alias": "oic.r.sensor.carbondioxide",
557       "x-to-ocf": [
558         "ocf.r.sensor.carbondioxide.precision = Precision"
559       ],
560       "x-from-ocf": [
561         "N/A"
562       ]
563     },
564   },
565   "Scale": {
566     "type" : "Integer",
567     "description": " indicate what scale is used for the actual sensor reading ",
568     "x-ocf-conversion": {
569       "x-ocf-alias": "oic.r.sensor.carbondioxide",
570       "x-to-ocf": [
571         "N/A"
572       ],
573       "x-from-ocf": [
574         "Scale = ppm (0x00)"
575       ]
576     },
577   },
578   "Size": {
579     "type" : "enum",
580     "description": " indicate the length in bytes of the Sensor Value field ",
581     "x-ocf-conversion": {
582       "x-ocf-alias": "oic.r.sensor.carbondioxide",
583       "x-to-ocf": [
584         "N/A"
585       ],
586       "x-from-ocf": [
587         "N/A"
588       ]
589     },
590   },
591   "Sensor Value": {
592     "type" : "array",
593     "description": " specify the value of the actual sensor reading ",
594     "x-ocf-conversion": {
595       "x-ocf-alias": "oic.r.sensor.carbondioxide",
596       "x-to-ocf": [
597         "ocf.r.sensor.carbondioxide.value = true",
598         "ocf.r.sensor.carbondioxide.measurement = Sensor Value"
599       ],
600       "x-from-ocf": [
601         "N/A"
602       ]
603     },
604   },
605 },
606 },
607 },
608 "type": "object",
609 "allof": [
610   {"$ref": "#/definitions/zwave.operation.multilevelsensorcommandclasscarbondioxide"}
611 ]
```



```

611     ],
612     "required": ["Sensor Type", "Precision", "Scale", "Size", "Sensor Value"]
613 }
614

```

615 8.5 Multilevel Sensor Command Class Carbon Monoxide

616 8.5.1 Derived model

617 The derived model: zwave.operation.multilevelsensorcommandclasscarbonmonoxide

618 8.5.2 Property definition

619 **Table 12 The property mapping for**
620 **zwave.operation.multilevelsensorcommandclasscarbonmonoxide**

ZWave Property name	OCF Resource	To OCF	From OCF
Precision	oic.r.sensor.carbonmonoxide	ocf.r.sensor.carbonmonoxide.precision = Precision	N/A
Size	oic.r.sensor.carbonmonoxide	N/A	N/A
Scale	oic.r.sensor.carbonmonoxide	N/A	Scale = ppm (0x01)
Sensor Value	oic.r.sensor.carbonmonoxide	ocf.r.sensor.carbonmonoxide.value = trueocf.r.sensor.carbonmonoxide.measurement = Sensor Value	N/A
Sensor Type	oic.r.sensor.carbonmonoxide	if Sensor Type = Carbon monoxide (CO) level, ocf.rt = oic.r.sensor.carbonmonoxide.	N/A

621 **Table 13 The properties of**
622 **zwave.operation.multilevelsensorcommandclasscarbonmonoxide**

ZWave name	Property	Type	Required	Description
Precision		Number	yes	indicate how many decimal places are included the Sensor Value field
Size		enum	yes	indicate the length in bytes of the Sensor Value field
Scale		Integer	yes	indicate what scale is used for the actual sensor reading
Sensor Value		array	yes	specify the value of the actual sensor reading
Sensor Type		Integer	yes	specify the carbon monoxide sensor type of the actual sensor reading

623 8.5.3 Derived model definition

```

624 {
625   "id":
626   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.multilevelsensorcommandclasscarbon
627   monoxide.json#",

```



```
628 "$schema": "http://json-schema.org/draft-04/schema#",
629 "description" : "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
630 "title": "Multilevel Sensor Command Class Carbon Monoxide",
631 "definitions": {
632   "zwave.operation.multilevelsensorcommandclasscarbonmonoxide": {
633     "type": "object",
634     "properties": {
635       "Sensor Type": {
636         "type" : "Integer",
637         "description": " specify the carbon monoxidesensor type of the actual sensor reading ",
638         "x-ocf-conversion": {
639           "x-ocf-alias": "oic.r.sensor.carbonmonoxide",
640           "x-to-ocf": [
641             "if Sensor Type = Carbon monoxide (CO) level, ocf.rt = oic.r.sensor.carbonmonoxide."
642           ],
643           "x-from-ocf": [
644             "N/A"
645           ]
646         }
647       },
648       "Precision": {
649         "type" : "Number",
650         "description": " indicate how many decimal places are included the Sensor Value field ",
651         "x-ocf-conversion": {
652           "x-ocf-alias": "oic.r.sensor.carbonmonoxide",
653           "x-to-ocf": [
654             "ocf.r.sensor.carbonmonoxide.precision = Precision"
655           ],
656           "x-from-ocf": [
657             "N/A"
658           ]
659         }
660       },
661       "Scale": {
662         "type" : "Integer",
663         "description": " indicate what scale is used for the actual sensor reading ",
664         "x-ocf-conversion": {
665           "x-ocf-alias": "oic.r.sensor.carbonmonoxide",
666           "x-to-ocf": [
667             "N/A"
668           ],
669           "x-from-ocf": [
670             "Scale = ppm (0x01)"
671           ]
672         }
673       },
674       "Size": {
675         "type" : "enum",
676         "description": " indicate the length in bytes of the Sensor Value field ",
677         "x-ocf-conversion": {
678           "x-ocf-alias": "oic.r.sensor.carbonmonoxide",
679           "x-to-ocf": [
680             "N/A"
681           ],
682           "x-from-ocf": [
683             "N/A"
684           ]
685         }
686       },
687       "Sensor Value": {
688         "type" : "array",
689         "description": " specify the value of the actual sensor reading ",
690         "x-ocf-conversion": {
691           "x-ocf-alias": "oic.r.sensor.carbonmonoxide",
692           "x-to-ocf": [
693             "ocf.r.sensor.carbonmonoxide.value = true",
694             "ocf.r.sensor.carbonmonoxide.measurement = Sensor Value"
695           ],
696           "x-from-ocf": [
697             "N/A"
698           ]
699         }
700     }
701   }
702 }
```



```

699     }
700   }
701 }
702 }
703 },
704 "type": "object",
705 "allOf": [
706   {"$ref": "#/definitions/zwave.operation.multilevelsensorcommandclasscarbonmonoxide"}
707 ],
708 "required": ["Sensor Type", "Precision", "Scale", "Size", "Sensor Value"]
709 }
710

```

711 **8.6 Multilevel Sensor Command Class Smoke Density**

712 **8.6.1 Derived model**

713 The derived model: zwave.operation.multilevelsensorcommandclasssmokedensity

714 **8.6.2 Property definition**

715 **Table 14 The property mapping for**
 716 **zwave.operation.multilevelsensorcommandclasssmokedensity**

ZWave Property name	OCF Resource	To OCF	From OCF
Sensor Value	oic.r.sensor.smoke	ocf.r.sensor.smoke.value = trueocf.r.sensor.smoke.measurement = Sensor Value	N/A
Size	oic.r.sensor.smoke	N/A	N/A
Sensor Type	oic.r.sensor.smoke	if Sensor Type = Smoke density, ocf.rt = oic.r.sensor.smoke.	N/A
Precision	oic.r.sensor.smoke	ocf.r.sensor.smoke.precision = Precision	N/A
Scale	oic.r.sensor.smoke	N/A	Scale = percent (0x00)

717 **Table 15 The properties of zwave.operation.multilevelsensorcommandclasssmokedensity**

ZWave name	Property	Type	Required	Description
Sensor Value		array	yes	specify the value of the actual sensor reading
Size		enum	yes	indicate the length in bytes of the Sensor Value field
Sensor Type		Integer	yes	specify the smoke density sensor type of the actual sensor reading
Precision		Number	yes	indicate how many decimal places are included the Sensor Value field
Scale		Integer	yes	indicate what scale is used for the actual sensor reading

718 **8.6.3 Derived model definition**

```

719 {
720   "id":

```



```
721 "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.multilevelsensorcommandclasssmoked
722 ensity.json#",
723 "$schema": "http://json-schema.org/draft-04/schema#",
724 "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
725 "title": "Multilevel Sensor Command Class Smoke Density",
726 "definitions": {
727   "zwave.operation.multilevelsensorcommandclasssmokedensity": {
728     "type": "object",
729     "properties": {
730       "Sensor Type": {
731         "type": "Integer",
732         "description": " specify the smoke density sensor type of the actual sensor reading ",
733         "x-ocf-conversion": {
734           "x-ocf-alias": "oic.r.sensor.smoke",
735           "x-to-ocf": [
736             "if Sensor Type = Smoke density, ocf.rt = oic.r.sensor.smoke."
737           ],
738           "x-from-ocf": [
739             "N/A"
740           ]
741         }
742       },
743       "Precision": {
744         "type": "Number",
745         "description": " indicate how many decimal places are included the Sensor Value field ",
746         "x-ocf-conversion": {
747           "x-ocf-alias": "oic.r.sensor.smoke",
748           "x-to-ocf": [
749             "ocf.r.sensor.smoke.precision = Precision"
750           ],
751           "x-from-ocf": [
752             "N/A"
753           ]
754         }
755       },
756       "Scale": {
757         "type": "Integer",
758         "description": " indicate what scale is used for the actual sensor reading ",
759         "x-ocf-conversion": {
760           "x-ocf-alias": "oic.r.sensor.smoke",
761           "x-to-ocf": [
762             "N/A"
763           ],
764           "x-from-ocf": [
765             "Scale = percent (0x00)"
766           ]
767         }
768       },
769       "Size": {
770         "type": "enum",
771         "description": " indicate the length in bytes of the Sensor Value field ",
772         "x-ocf-conversion": {
773           "x-ocf-alias": "oic.r.sensor.smoke",
774           "x-to-ocf": [
775             "N/A"
776           ],
777           "x-from-ocf": [
778             "N/A"
779           ]
780         }
781       },
782       "Sensor Value": {
783         "type": "array",
784         "description": " specify the value of the actual sensor reading ",
785         "x-ocf-conversion": {
786           "x-ocf-alias": "oic.r.sensor.smoke",
787           "x-to-ocf": [
788             "ocf.r.sensor.smoke.value = true",
789             "ocf.r.sensor.smoke.measurement = Sensor Value"
790           ],
791           "x-from-ocf": [
```



```

792         "N/A"
793     ]
794 }
795 }
796 }
797 }
798 },
799 "type": "object",
800 "allOf": [
801   {"$ref": "#/definitions/zwave.operation.multilevelsensorcommandclasssmokedensity"}
802 ],
803 "required": ["Sensor Type", "Precision", "Scale", "Size", "Sensor Value"]
804 }
805

```

8.7 Multilevel Sensor Command Class Water Flow

8.7.1 Derived model

The derived model: `zwave.operation.multilevelsensorcommandclasswaterflow`

8.7.2 Property definition

Table 16 The property mapping for `zwave.operation.multilevelsensorcommandclasswaterflow`

ZWave Property name	OCF Resource	To OCF	From OCF
Sensor Type	<code>oic.r.sensor.water</code>	if Sensor Type = Water flow, <code>ocf.rt = oic.r.sensor.water.</code>	N/A
Scale	<code>oic.r.sensor.water</code>	N/A	Scale = litre/hr (0x00)
Sensor Value	<code>oic.r.sensor.water</code>	<code>ocf.r.sensor.water.value = true</code> <code>ocf.r.sensor.water.measurement = Sensor Value</code>	N/A
Precision	<code>oic.r.sensor.water</code>	<code>ocf.r.sensor.water.precision = Precision</code>	N/A
Size	<code>oic.r.sensor.water</code>	N/A	N/A

Table 17 The properties of `zwave.operation.multilevelsensorcommandclasswaterflow`

ZWave Property name	Type	Required	Description
Sensor Type	Integer	yes	specify the water flow sensor type of the actual sensor reading
Scale	Integer	yes	indicate what scale is used for the actual sensor reading
Sensor Value	array	yes	specify the value of the actual sensor reading
Precision	Number	yes	indicate how many decimal places are included the Sensor Value field
Size	enum	yes	indicate the length in bytes of the Sensor Value field



813 8.7.3 Derived model definition

```
814 {
815   "id":
816   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.multilevelsensorcommandclasswaterf
817   low.json#",
818   "$schema": "http://json-schema.org/draft-04/schema#",
819   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
820   "title": "Multilevel Sensor Command Class Water Flow",
821   "definitions": {
822     "zwave.operation.multilevelsensorcommandclasswaterflow": {
823       "type": "object",
824       "properties": {
825         "Sensor Type": {
826           "type": "Integer",
827           "description": " specify the water flow sensor type of the actual sensor reading ",
828           "x-ocf-conversion": {
829             "x-ocf-alias": "oic.r.sensor.water",
830             "x-to-ocf": [
831               "if Sensor Type = Water flow, ocf.rt = oic.r.sensor.water."
832             ],
833             "x-from-ocf": [
834               "N/A"
835             ]
836           }
837         },
838         "Precision": {
839           "type": "Number",
840           "description": " indicate how many decimal places are included the Sensor Value field ",
841           "x-ocf-conversion": {
842             "x-ocf-alias": "oic.r.sensor.water",
843             "x-to-ocf": [
844               "ocf.r.sensor.water.precision = Precision"
845             ],
846             "x-from-ocf": [
847               "N/A"
848             ]
849           }
850         },
851         "Scale": {
852           "type": "Integer",
853           "description": " indicate what scale is used for the actual sensor reading ",
854           "x-ocf-conversion": {
855             "x-ocf-alias": "oic.r.sensor.water",
856             "x-to-ocf": [
857               "N/A"
858             ],
859             "x-from-ocf": [
860               "Scale = litre/hr (0x00)"
861             ]
862           }
863         },
864         "Size": {
865           "type": "enum",
866           "description": " indicate the length in bytes of the Sensor Value field ",
867           "x-ocf-conversion": {
868             "x-ocf-alias": "oic.r.sensor.water",
869             "x-to-ocf": [
870               "N/A"
871             ],
872             "x-from-ocf": [
873               "N/A"
874             ]
875           }
876         },
877         "Sensor Value": {
878           "type": "array",
879           "description": " specify the value of the actual sensor reading ",
880           "x-ocf-conversion": {
881             "x-ocf-alias": "oic.r.sensor.water",
882             "x-to-ocf": [
```




```

883         "ocf.r.sensor.water.value = true",
884         "ocf.r.sensor.water.measurement = Sensor Value"
885     ],
886     "x-from-ocf": [
887         "N/A"
888     ]
889 }
890 }
891 }
892 }
893 },
894 "type": "object",
895 "allOf": [
896     {"$ref": "#/definitions/zwave.operation.multilevelsensorcommandclasswaterflow"}
897 ],
898 "required": ["Sensor Type", "Precision", "Scale", "Size", "Sensor Value"]
899 }
900

```

901 8.8 Multilevel Switch Command Class

902 8.8.1 Derived model

903 The derived model: zwave.operation.multilevelswitchcommandclass

904 8.8.2 Property definition

905 **Table 18 The property mapping for zwave.operation.multilevelswitchcommandclass**

ZWave Property name	OCF Resource	To OCF	From OCF
Value	oic.r.switch.binary, oic.r.light.dimming	if value = 0, ocf.rt = oic.r.switch.binary & ocf.r.switch.binary.value = false otherwise: ocf.rt = oic.r.light.dimming; ocf.r.light.dimming.dimmingSetting = value	value = dimmingSetting if ocf.rt = oic.r.switch.binary, value = ocf.r.switch.binary.value if ocf.rt = oic.r.light.dimming, value = dimmingSetting

906 **Table 19 The properties of zwave.operation.multilevelswitchcommandclass**

ZWave name	Property	Type	Required	Description
Value		integer, boolean	yes	multilevel value in a supporting device

907 8.8.3 Derived model definition

```

908 {
909     "id":
910     "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.multilevelswitchcommandclass.json#",
911     "schema": "http://json-schema.org/draft-04/schema#",
912     "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
913     "title": "Multilevel Switch Command Class",
914     "definitions": {
915         "zwave.operation.multilevelswitchcommandclass": {
916             "type": "object",
917             "properties": {
918                 "Value": {
919                     "type": "integer, boolean",
920                     "description": "multilevel value in a supporting device",
921                     "x-ocf-conversion": {
922                         "x-ocf-alias": "oic.r.switch.binary, oic.r.light.dimming",
923                         "x-to-ocf": [
924                             "if value = 0, ocf.rt = oic.r.switch.binary & ocf.r.switch.binary.value = false",
925

```



```

926     "otherwise: ocf.rt = oic.r.light.dimming; ocf.r.light.dimming.dimmingSetting = value"
927   ],
928   "x-from-ocf": [
929     "value = dimmingSetting",
930     "if ocf.rt = oic.r.switch.binary, value = ocf.r.switch.binary.value",
931     "if ocf.rt = oic.r.light.dimming, value = dimmingSetting"
932   ]
933 }
934 }
935 }
936 }
937 },
938 "type": "object",
939 "allOf": [
940   {"$ref": "#/definitions/zwave.operation.multilevelswitchcommandclass"}
941 ],
942 "required": ["Value"]
943 }

```

8.9 Notification Command Class

8.9.1 Derived model

The derived model: `zwave.operation.notificationcommandclass`

8.9.2 Property definition

Table 20 The property mapping for `zwave.operation.notificationcommandclass`

ZWave Property name	OCF Resource	To OCF	From OCF
V1 Alarm Level	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	N/A	N/A
Notification Status	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	Value = Notification Status	N/A
V1 Alarm Type	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	N/A	N/A
Event:State Parameters Length	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	ocf.event:stateparameterslength = Event:State Parameters Length	N/A
Sequence	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	ocf.sequence = Sequence	N/A
Notification Type	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	if Notification Type = Smoke Alarm, ocf.rt = oic.r.sensor.smoke. if Notification Type = CO Alarm, ocf.rt = oic.r.sensor.carbonmonoxide. if Notification Type = CO2 Alarm, ocf.rt = oic.r.sensor.carbondioxide. if	N/A



		Notification Type = Water Alarm, ocf.rt = oic.r.sensor.water.	
Event:State Parameter	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	ocf.event:stateparameter Event:State Parameter =	N/A
Sequence Number	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	ocf.sequencenumber Sequence Number =	N/A
Notification Event:State	oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide, oic.r.sensor.smoke, oic.r.sensor.water	Value = Notification Event:State	N/A

949

Table 21 The properties of zwave.operation.notificationcommandclass

ZWave name	Property	Type	Required	Description
V1 Alarm Level		Integer	yes	product manual specific
Notification Status		Integer	yes	advertise the status of the Notification Type
V1 Alarm Type		Integer	yes	depends on the V1 Alarm field advertised in the Alarm Type Supported Report Command
Event:State Parameters Length		number	yes	advertise the length in bytes of the Event / State Parameters field
Sequence		boolean	yes	advertise the presence of the Sequence Number field
Notification Type		Integer	yes	specify a Notification Type
Event:State Parameter		Integer	no	specify associated parameters to a Notification
Sequence Number		number	no	advertise a sequence number for the actual Notification
Notification Event:State		Integer	yes	specify a Notification Event/State for the advertised Notification Type

950

8.9.3 Derived model definition

951
952
953
954
955
956
957
958
959

```
{
  "id":
  "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.notificationcommandclass.json#",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
  "title": "Notification Command Class",
  "definitions": {
    "zwave.operation.notificationcommandclass": {
      "type": "object",
```



```
960     "properties": {
961         "V1 Alarm Type": {
962             "type" : "Integer",
963             "description": "depends on the V1 Alarm field advertised in the Alarm Type Supported
964 Report Command",
965             "x-ocf-conversion": {
966                 "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
967 oic.r.sensor.smoke, oic.r.sensor.water",
968                 "x-to-ocf": [
969                     "N/A"
970                 ],
971                 "x-from-ocf": [
972                     "N/A"
973                 ]
974             }
975         },
976         "V1 Alarm Level": {
977             "type" : "Integer",
978             "description": "product manual specific",
979             "x-ocf-conversion": {
980                 "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
981 oic.r.sensor.smoke, oic.r.sensor.water",
982                 "x-to-ocf": [
983                     "N/A"
984                 ],
985                 "x-from-ocf": [
986                     "N/A"
987                 ]
988             }
989         },
990         "Notification Status": {
991             "type" : "Integer",
992             "description": "advertise the status of the Notification Type",
993             "x-ocf-conversion": {
994                 "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
995 oic.r.sensor.smoke, oic.r.sensor.water",
996                 "x-to-ocf": [
997                     "Value = Notification Status"
998                 ],
999                 "x-from-ocf": [
1000                     "N/A"
1001                 ]
1002             }
1003         },
1004         "Notification Type": {
1005             "type" : "Integer",
1006             "description": " specify a Notification Type ",
1007             "x-ocf-conversion": {
1008                 "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1009 oic.r.sensor.smoke, oic.r.sensor.water",
1010                 "x-to-ocf": [
1011                     "if Notification Type = Smoke Alarm, ocf.rt = oic.r.sensor.smoke.",
1012                     "if Notification Type = CO Alarm, ocf.rt = oic.r.sensor.carbonmonoxide.",
1013                     "if Notification Type = CO2 Alarm, ocf.rt = oic.r.sensor.carbondioxide.",
1014                     "if Notification Type = Water Alarm, ocf.rt = oic.r.sensor.water."
1015                 ],
1016                 "x-from-ocf": [
1017                     "N/A"
1018                 ]
1019             }
1020         },
1021         "Notification Event:State": {
1022             "type" : "Integer",
1023             "description": "specify a Notification Event/State for the advertised Notification Type",
1024             "x-ocf-conversion": {
1025                 "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1026 oic.r.sensor.smoke, oic.r.sensor.water",
1027                 "x-to-ocf": [
1028                     "Value = Notification Event:State"
1029                 ],
1030                 "x-from-ocf": [
```



```
1031         "N/A"
1032     ]
1033 }
1034 },
1035 "Sequence": {
1036     "type" : "boolean",
1037     "description": "advertise the presence of the Sequence Number field",
1038     "x-ocf-conversion": {
1039         "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1040 oic.r.sensor.smoke, oic.r.sensor.water",
1041         "x-to-ocf": [
1042             "ocf.sequence = Sequence"
1043         ],
1044         "x-from-ocf": [
1045             "N/A"
1046         ]
1047     }
1048 },
1049 "Event:State Parameters Length": {
1050     "type" : "number",
1051     "description": "advertise the length in bytes of the Event / State Parameters field",
1052     "x-ocf-conversion": {
1053         "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1054 oic.r.sensor.smoke, oic.r.sensor.water",
1055         "x-to-ocf": [
1056             "ocf.event:stateparameterslength = Event:State Parameters Length"
1057         ],
1058         "x-from-ocf": [
1059             "N/A"
1060         ]
1061     }
1062 },
1063 "Event:State Parameter": {
1064     "type" : "Integer",
1065     "description": "specify associated parameters to a Notification",
1066     "x-ocf-conversion": {
1067         "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1068 oic.r.sensor.smoke, oic.r.sensor.water",
1069         "x-to-ocf": [
1070             "ocf.event:stateparameter = Event:State Parameter"
1071         ],
1072         "x-from-ocf": [
1073             "N/A"
1074         ]
1075     }
1076 },
1077 "Sequence Number": {
1078     "type" : "number",
1079     "description": "advertise a sequence number for the actual Notification",
1080     "x-ocf-conversion": {
1081         "x-ocf-alias": "oic.r.sensor.carbondioxide, oic.r.sensor.carbonmonoxide,
1082 oic.r.sensor.smoke, oic.r.sensor.water",
1083         "x-to-ocf": [
1084             "ocf.sequencenumber = Sequence Number"
1085         ],
1086         "x-from-ocf": [
1087             "N/A"
1088         ]
1089     }
1090 }
1091 }
1092 }
1093 },
1094 "type": "object",
1095 "allof": [
1096     {"$ref": "#/definitions/zwave.operation.notificationcommandclass"}
1097 ],
1098 "required": ["V1 Alarm Type", "V1 Alarm Level", "Notification Status", "Notification Type",
1099 "Notification Event:State", "Sequence", "Event:State Parameters Length"]
1100 }
```



1101 **8.10 User Code Command Class**

1102 **8.10.1 Derived model**

1103 The derived model: zwave.operation.usercodecommandclass

1104 **8.10.2 Property definition**

1105 **Table 22 The property mapping for zwave.operation.usercodecommandclass**

ZWave Property name	OCF Resource	To OCF	From OCF
User Identifier	oic.r.lock.code	Used as an index in the lock code array. It is defined in ZWave as 0..255 (8 bit field).	useridentifier = oic.r.lock.code.lockCodeList[arrayIndex]
User ID Status	oic.r.lock.code	N/A	User ID Status = 0x01
lockCodeList	oic.r.lock.code	User Identifier = ZWave Command Class User Identifier oic.r.lock.code.lockCodeList[User Identifier] = User Code	User Identifier = locally persisted ZWave Command Class User Identifier associated with this Resource User Code = oic.r.lock.code.lockCodeList[User Identifier]

1106 **Table 23 The properties of zwave.operation.usercodecommandclass**

ZWave Property name	Type	Required	Description
User Identifier	Number	yes	specify the actual User Identifier
User ID Status	Integer	yes	indicates the status of the User Identifier
lockCodeList	array	no	advertise the User Code to be set for the User Identifier

1107 **8.10.3 Derived model definition**

```

1108 {
1109   "id":
1110   "http://openinterconnect.org/zwavemapping/schemas/zwave.operation.usercodecommandclass.json#",
1111   "$schema": "http://json-schema.org/draft-04/schema#",
1112   "description": "Copyright (c) 2018 Open Connectivity Foundation, Inc. All rights reserved.",
1113   "title": "User Code Command Class",
1114   "definitions": {
1115     "zwave.operation.usercodecommandclass": {
1116       "type": "object",
1117       "properties": {
1118         "User Identifier": {
1119           "type": "Number",
1120           "description": "specify the actual User Identifier",
1121           "x-ocf-conversion": {
1122             "x-ocf-alias": "oic.r.lock.code",
1123             "x-to-ocf": [
1124               "Used as an index in the lock code array. It is defined in ZWave as 0..255 (8 bit
1125 field)."
1126             ],
1127             "x-from-ocf": [
1128               "useridentifier = oic.r.lock.code.lockCodeList[arrayIndex]"
1129             ]
1130           }
1131         },
1132         "User ID Status": {
1133           "type": "Integer",

```



```
1134         "description": "indicates the status of the User Identifier",
1135         "x-ocf-conversion": {
1136             "x-ocf-alias": "oic.r.lock.code",
1137             "x-to-ocf": [
1138                 "N/A "
1139             ],
1140             "x-from-ocf": [
1141                 "User ID Status = 0x01"
1142             ]
1143         }
1144     },
1145     "lockCodeList": {
1146         "type": "array",
1147         "description": "advertise the User Code to be set for the User Identifier",
1148         "x-ocf-conversion": {
1149             "x-ocf-alias": "oic.r.lock.code",
1150             "x-to-ocf": [
1151                 "User Identifier = ZWave Command Class User Identifier",
1152                 "oic.r.lock.code.lockCodeList[User Identifier] = User Code"
1153             ],
1154             "x-from-ocf": [
1155                 "User Identifier = locally persisted ZWave Command Class User Identifier associated
1156 with this Resource",
1157                 "User Code = oic.r.lock.code.lockCodeList[User Identifier]"
1158             ]
1159         }
1160     }
1161 }
1162 }
1163 },
1164 "type": "object",
1165 "allOf": [
1166     {"$ref": "#/definitions/zwave.operation.doorlockoperationcommandclass"}
1167 ],
1168 "required": ["User Identifier", "User ID Status", "User Code"]
1169 }
```