

OCF Bridging Specification

VERSION 2.0.4 | July 2019



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

Copyright Open Connectivity Foundation, Inc. © 2019.
All Rights Reserved.

1 **LEGAL DISCLAIMER**

2
3 NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND
4 OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY
5 INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR
6 DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED
7 ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW,
8 THE AUTHORS AND DEVELOPERS OF THIS SPECIFICATION HEREBY DISCLAIM ALL OTHER
9 WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT
10 COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
11 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN CONNECTIVITY
12 FOUNDATION, INC. FURTHER DISCLAIMS ANY AND ALL WARRANTIES OF NON-
13 INFRINGEMENT, ACCURACY OR LACK OF VIRUSES.

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

16 Copyright © 2017-2019 Open Connectivity Foundation, Inc. All rights reserved.

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

CONTENTS

21	1	Scope.....	1
22	2	Normative references	1
23	3	Terms, definitions, and abbreviated terms	2
24	3.1	Terms and definitions.....	2
25	3.2	Abbreviated terms.....	5
26	4	Document conventions and organization.....	7
27	4.1	Conventions.....	7
28	4.2	Notation.....	7
29	5	Bridge Platform.....	8
30	5.1	Introduction.....	8
31	5.2	Symmetric vs. asymmetric bridging	9
32	5.3	General requirements	11
33	5.3.1	For Asymmetric Bridging.....	11
34	5.3.2	For Symmetric Bridging	11
35	5.4	VOD List	11
36	5.5	Resource discovery	11
37	5.6	"Deep translation" vs. "on-the-fly"	17
38	5.7	Security	17
39	6	AllJoyn translation	17
40	6.1	Operational scenarios	17
41	6.2	Requirements specific to an AllJoyn Bridging Function	17
42	6.2.1	Introduction	17
43	6.2.2	Use of introspection.....	18
44	6.2.3	Stability and loss of data.....	18
45	6.2.4	Exposing AllJoyn producer devices to OCF clients.....	18
46	6.2.5	Exposing OCF resources to AllJoyn consumer applications	27
47	6.2.6	Security.....	34
48	6.3	On-the-Fly Translation from D-Bus and OCF payloads	34
49	6.3.1	Introduction	34
50	6.3.2	Translation without aid of introspection.....	34
51	6.3.3	Translation with aid of introspection.....	40
52	7	oneM2M Translation	45
53	7.1	Operational Scenarios	45
54	7.2	Enabling oneM2M Application access to OCF Servers	46
55	7.3	Enabling OCF Client access to oneM2M Devices	46
56	7.4	On-the-fly Translation	46
57	8	Zigbee Translation.....	46
58	8.1	Operational Scenarios	46
59	8.2	Requirements specific to Zigbee Bridging Function	47

60	8.2.1	Requirements specific to Zigbee	47
61	8.2.2	Exposing Zigbee 3.0 Servers to OCF Clients	47
62	8.2.3	Translation for well-defined set	49
63	8.2.4	Exposing a Zigbee 3.0 Server as a Virtual OCF Server	50
64	9	Device type definitions	56
65	10	Resource type definitions	56
66	10.1	List of resource types	56
67	10.2	AllJoynObject	57
68	10.2.1	Introduction	57
69	10.2.2	Example URI	57
70	10.2.3	Resource type	57
71	10.2.4	OpenAPI 2.0 definition	57
72	10.2.5	Property definition	61
73	10.2.6	CRUDN behaviour	61
74	10.3	SecureMode	62
75	10.3.1	Introduction	62
76	10.3.2	Example URI	62
77	10.3.3	Resource type	62
78	10.3.4	OpenAPI 2.0 definition	62
79	10.3.5	Property definition	64
80	10.3.6	CRUDN behaviour	64
81	10.4	VOD List	64
82	10.4.1	Introduction	64
83	10.4.2	Example URI	64
84	10.4.3	Resource type	65
85	10.4.4	OpenAPI 2.0 definition	65
86	10.4.5	Property definition	67
87	10.4.6	CRUDN behaviour	67
88			
89			

90

Figures

91	Figure 1 – Bridge Platform components	8
92	Figure 2 – Schematic overview of a Bridge Platform bridging non-OCF devices	9
93	Figure 3 – Asymmetric server bridge.....	10
94	Figure 4 – Asymmetric client bridge	10
95	Figure 5 – /oic/res example responses.....	17
96	Figure 6 – Payload Chain.....	18
97	Figure 7 – OCF Zigbee Bridge Platform and Components	47
98		

Tables

100	Table 1 – AllJoyn Bridging Function Interaction List.....	18
101	Table 2 – AllJoyn to OCF Name Examples.....	20
102	Table 3 – oic.wk.d resource type definition	21
103	Table 4 – oic.wk.con resource type definition.....	23
104	Table 5 – oic.wk.p resource type definition	25
105	Table 6 – oic.wk.con.p resource type definition.....	26
106	Table 7 – Example name mapping	28
107	Table 8 – AllJoyn about data fields	29
108	Table 9 – AllJoyn configuration data fields	32
109	Table 10 – Boolean translation	34
110	Table 11 – Numeric type translation, D-Bus to JSON	34
111	Table 12 – Numeric type translation, JSON to D-Bus	35
112	Table 13 – Text string translation.....	35
113	Table 14 – Byte array translation	35
114	Table 15 – D-Bus variant translation	36
115	Table 16 – D-Bus object path translation	36
116	Table 17 – D-Bus structure translation	36
117	Table 18 – Byte array translation	37
118	Table 19 – Other array translation	37
119	Table 20 – JSON array translation	37
120	Table 21 – D-Bus dictionary translation.....	37
121	Table 22 – Non-translation types	38
122	Table 23 – D-Bus to JSON translation examples.....	39
123	Table 24 – JSON to D-Bus translation examples.....	40
124	Table 25 – JSON type to D-Bus type translation	42
125	Table 26 – D-Bus type to JSON type translation	42
126	Table 27 – Text string translation.....	43
127	Table 28 – JSON UUID string translation	43
128	Table 29 – D-Bus variant translation	43
129	Table 30 – D-Bus object path translation	43
130	Table 31 – Mapping from AllJoyn using introspection.....	44
131	Table 32 – Mapping from CBOR using introspection	45
132	Table 33 – Translation Rule between Zigbee and OCF Data Models.....	48
133	Table 34 – Zigbee to OCF Mapping Example (Color Temperature Light)	48
134	Table 35 – Zigbee 3.0 Device & Cluster – OCF Device & Resource mapping	49
135	Table 36 – "oic.wk.p" Resource Type mapping.....	50
136	Table 37 – "oic.wk.d" Resource Type mapping.....	52

137	Table 38 – "oic.wk.con" Resource Type mapping	55
138	Table 39 – Device type definitions	56
139	Table 40 – Alphabetical list of resource types	56
140	Table 41 – The Property definitions of the Resource with type "rt" = "oic.r.alljoynobject,	
141	oic.wk.col".	61
142	Table 42 – The CRUDN operations of the Resource with type "rt" = "oic.r.alljoynobject,	
143	oic.wk.col".	61
144	Table 43 – The Property definitions of the Resource with type "rt" = "oic.r.securemode".	64
145	Table 44 – The CRUDN operations of the Resource with type "rt" = "oic.r.securemode".....	64
146	Table 45 – The Property definitions of the Resource with type "rt" = "oic.r.vodlist".	67
147	Table 46 – The CRUDN operations of the Resource with type "rt" = "oic.r.vodlist".	67
148		

149 **1 Scope**

150 This document specifies a framework for translation between OCF Devices and other ecosystems,
151 and specifies the behaviour of a Bridging Function that exposes servers in non-OCF ecosystem to
152 OCF Clients and/or exposes OCF Servers to clients in non-OCF ecosystem. Translation per
153 specific Device is left to other documents (deep translation). This document provides generic
154 requirements that apply unless overridden by a more specific document.

155 **2 Normative references**

156 The following documents are referred to in the text in such a way that some or all of their content
157 constitutes requirements of this document. For dated references, only the edition cited applies. For
158 undated references, the latest edition of the referenced document (including any amendments)
159 applies.

160 AllJoyn About Interface Specification, *About Feature Interface Definitions*, Version 14.12
161 <https://allseenalliance.org/framework/documentation/learn/core/about-announcement/interface>

162 AllJoyn Configuration Interface Specification, *Configuration Interface Definition*, Version 14.12
163 <https://allseenalliance.org/framework/documentation/learn/core/configuration/interface>

164 D-Bus Specification, *D-Bus Specification*
165 <https://dbus.freedesktop.org/doc/dbus-specification.html>

166 IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, August 2008
167 <http://ieeexplore.ieee.org/servlet/opac?punumber=4610933>

168 IETF RFC 4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005
169 <https://www.rfc-editor.org/info/rfc4122>

170 IETF RF 4648, *The Base16, Base32 and Base64 Data Encodings*, October 2006
171 <https://www.rfc-editor.org/info/rfc4648>

172 IETF RFC 6973, *Privacy Considerations for Internet Protocols*, July 2013
173 <https://www.rfc-editor.org/info/rfc6973>

174 IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014
175 <https://www.rfc-editor.org/info/rfc7159>

176 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)
177 Specification -- Part 1: Core specification
178 <https://www.iso.org/standard/53238.html>
179 Latest version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf

180 ISO/IEC 30118-2:2018 Information technology -- Open Connectivity Foundation (OCF)
181 Specification -- Part 2: Security specification
182 <https://www.iso.org/standard/74239.html>
183 Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

184 ISO/IEC 30118-6:2018 Information technology -- Open Connectivity Foundation (OCF)
185 Specification -- Part 6: Resource to AllJoyn interface mapping specification
186 <https://www.iso.org/standard/74243.html>
187 Latest version available at:
188 https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping.pdf

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

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

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

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

198 OCF Resource to oneM2M Module Class Mapping, *Open Connectivity Foundation Resource to*
199 *oneM2M Module Class Mapping Specification*, version 2.0.2

200 Available at:

201 [https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specifi](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.0.2.pdf)
202 [cation_v2.0.2.pdf](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.0.2.pdf)

203 Latest version available at:

204 [https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specifi](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification.pdf)
205 [cation.pdf](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification.pdf)

206 OCF Resource to Zigbee Cluster Mapping, *Open Connectivity Foundation Resource to Zigbee*
207 *Cluster Mapping Specification*, version 2.0.3

208 Available at:

209 [https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.](https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.0.3.pdf)
210 [0.3.pdf](https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.0.3.pdf)

211 Latest version available at:

212 https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification.pdf

213 Zigbee, *Zigbee Specification*, August 2015

214 <http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/>

215 Zigbee Cluster Library, *Zigbee Cluster Library Specification*, January 2016

216 <http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/>

217 **3 Terms, definitions, and abbreviated terms**

218 **3.1 Terms and definitions**

219 For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2018 and
220 the following apply.

221 ISO and IEC maintain terminological databases for use in standardization at the following
222 addresses:

223 – ISO Online browsing platform: available at <https://www.iso.org/obp>

224 – IEC Electropedia: available at <http://www.electropedia.org/>

225 **3.1.1**

226 **Asymmetric Client Bridge**

227 an asymmetric client bridge exposes another ecosystem clients into the OCF ecosystem as Virtual
228 OCF Clients (3.1.2). This is equivalent to exposing OCF Servers (3.1.15) into the other ecosystem.
229 How this is handled in each ecosystem is specified on a per ecosystem basis in this document.

230 **3.1.2**

231 **Asymmetric Server Bridge**

232 an asymmetric server bridge exposes another ecosystem devices into the OCF ecosystem as
233 Virtual OCF Servers (3.1.26). How this is handled in each ecosystem is specified on a per
234 ecosystem basis in this document.

235 **3.1.3**
236 **Bridge**
237 OCF Device that has a Device Type of "oic.d.bridge", provides information on the set of Virtual
238 OCF Devices (3.1.24) that are resident on the same Bridge Platform.

239 **3.1.4**
240 **Bridge Platform**
241 Entity on which the Bridge (3.1.2) and Virtual OCF Devices (3.1.25) are resident

242 **3.1.5**
243 **Bridged Client**
244 logical entity that accesses data via a Bridged Protocol (3.1.5). For example, an AllJoyn Consumer
245 application is a Bridged Client

246 **3.1.6**
247 **Bridged Device**
248 Bridged Client (3.1.3) or Bridged Server (3.1.8).

249 **3.1.7**
250 **Bridged Protocol**
251 another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

252 **3.1.8**
253 **Bridged Resource**
254 represents an artefact modelled and exposed by a Bridged Protocol (3.1.5), for example an AllJoyn
255 object is a Bridged Resource.

256 **3.1.9**
257 **Bridged Resource Type**
258 schema used with a Bridged Protocol (3.1.5), for example AllJoyn Interfaces are Bridged Resource
259 Types.

260 **3.1.10 Bridged Server**
261 logical entity that provides data via a Bridged Protocol (3.1.5), for example an AllJoyn Producer is
262 a Bridged Server. More than one Bridged Server can exist on the same physical platform.

263 **3.1.11**
264 **Bridging Function**
265 Logic resident on the Bridge Platform (3.1.4) that performs that protocol mapping between OCF
266 and the Bridged Protocol (3.1.7); a Bridge Platform (3.1.4) may contain multiple Bridging Functions
267 dependent on the number of Bridged Protocols (3.1.7) supported.

268 **3.1.12**
269 **OCF Bridge Device**
270 OCF Device (3.1.11) that can represent devices that exist on the network but communicate using
271 a Bridged Protocol (3.1.5) rather than OCF protocols.

272 **3.1.13**
273 **OCF Client**
274 logical entity that accesses an OCF Resource (3.1.12) on an OCF Server (3.1.15), which might be
275 a Virtual OCF Server (3.1.26) exposed by the OCF Bridge Device (3.1.9)

276 **3.1.14**
277 **OCF Device**
278 logical entity that assumes one or more OCF roles (OCF Client (3.1.10), OCF Server (3.1.15)). More
279 than one OCF Device can exist on the same physical platform.

280 **3.1.15**
281 **OCF Resource**
282 represents an artefact modelled and exposed by the OCF Framework

283 **3.1.16**
284 **OCF Resource Property**
285 significant aspect or notion including metadata that is exposed through the OCF Resource (3.1.12)

286 **3.1.17**
287 **OCF Resource Type**
288 OCF Resource Property (3.1.13) that represents the data type definition for the OCF Resource
289 (3.1.12)

290 **3.1.18**
291 **OCF Server**
292 logical entity with the role of providing resource state information and allowing remote control of its
293 resources

294 **3.1.19**
295 **oneM2M Application**
296 In an OCF-oneM2M asymmetric bridge environment, the oneM2M application represents the
297 oneM2M control point (i.e. client) being mapped to a virtual OCF client.

298 **3.1.20**
299 **Symmetric, Asymmetric Bridging**
300 in symmetric bridging, a bridge device exposes OCF Server(s) (3.1.15) to another ecosystem and
301 exposes other ecosystem's server(s) to OCF. In asymmetric bridging, a bridge device exposes
302 OCF Server(s) (3.1.15) to another ecosystem or exposes another ecosystem's server(s) to OCF,
303 but not both.

304 **3.1.21**
305 **Virtual Bridged Client**
306 logical representation of an OCF Client (3.1.10), which an OCF Bridge Device (3.1.9) exposes to
307 Bridged Servers (3.1.8).

308 **3.1.22**
309 **Virtual Bridged Server**
310 logical representation of an OCF Server (3.1.15), which an OCF Bridge Device (3.1.9) exposes to
311 Bridged Clients (3.1.3).

312 **3.1.23**
313 **Virtual OCF Client**
314 logical representation of a Bridged Client (3.1.3), which an OCF Bridge Device (3.1.9) exposes to
315 OCF Servers (3.1.15)

316 **3.1.24**
317 **Virtual OCF Device**
318 Virtual OCF Client (3.1.23) or Virtual OCF Server (3.1.26).

319 **3.1.25**
320 **Virtual OCF Resource**
321 logical representation of a Bridged Resource (3.1.6), which an OCF Bridge Device (3.1.9) exposes
322 to OCF Clients (3.1.10)

323 **3.1.26**
324 **Virtual OCF Server**
325 logical representation of a Bridged Server (3.1.8), which an OCF Bridge Device (3.1.9) exposes to
326 OCF Clients (3.1.10).

327 **3.1.27**
328 **Zigbee Attribute**
329 data entity which represents a physical quantity or state within Zigbee. This data is communicated
330 to other devices using commands.

331 **3.1.28**
332 **Zigbee Cluster**
333 one or more Zigbee Attributes (3.1.27), commands, behaviours, and dependencies, which supports
334 an independent utility or application function. The term may also be used for an implementation or
335 instance of such on an endpoint.

336 **3.1.29**
337 **Zigbee Server**
338 cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint.
339 Typically this interface supports all or most of the attributes of the cluster. A server cluster
340 communicates with a corresponding remote client cluster with the same identifier.

341 **3.1.30**
342 **Zigbee 3.0 Server**
343 Zigbee Server (3.1.29) which is built on Zigbee 3.0 stack

344 **3.1.31**
345 **Zigbee Client**
346 cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint.
347 Typically this interface sends commands that manipulate the attributes on the corresponding
348 Zigbee Server (3.1.29). A client cluster communicates with a corresponding remote server cluster
349 with the same identifier.

350 **3.1.32**
351 **Zigbee 3.0 Client**
352 Zigbee Client (3.1.31) which is built on Zigbee 3.0 stack

353 **3.1.33**
354 **Zigbee Device**
355 unique device identifier and a set of mandatory and optional clusters to be implemented on a single
356 Zigbee endpoint. The term may also be used for an implementation or instance on an endpoint. In
357 this document, the unique identifier of a Zigbee Device maps to an OCF Device Type.

358 **3.1.34**
359 **Zigbee 3.0 Device**
360 Zigbee Device (3.1.33) which is built on Zigbee 3.0 stack

361 **3.2 Abbreviated terms**

362 **3.2.1**
363 **CRUDN**
364 Create, Read, Update, Delete, and Notify

365 **3.2.2**
366 **CSV**
367 Comma separated value

369 4 Document conventions and organization

370 4.1 Conventions

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

375 4.2 Notation

376 In this document, features are described as required, recommended, allowed or DEPRECATED as
377 follows:

378 Required (or shall or mandatory).

- 379 – These basic features shall be implemented to comply with OIC Core Architecture. The phrases
380 "shall not", and "PROHIBITED" indicate behaviour that is prohibited, i.e. that if performed means
381 the implementation is not in compliance.

382 Recommended (or should).

- 383 – These features add functionality supported by OIC Core Architecture and should be
384 implemented. Recommended features take advantage of the capabilities OIC Core Architecture,
385 usually without imposing major increase of complexity. Notice that for compliance testing, if a
386 recommended feature is implemented, it shall meet the specified requirements to be in
387 compliance with these guidelines. Some recommended features could become requirements in
388 the future. The phrase "should not" indicates behaviour that is permitted but not recommended.

389 Allowed (or allowed).

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

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

395 Conditionally required (CR)

- 396 – The definition or behaviour depends on a condition. If the specified condition is met, then the
397 definition or behaviour is required. Otherwise the definition or behaviour is allowed as default
398 unless specifically defined as not allowed.

399 DEPRECATED

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

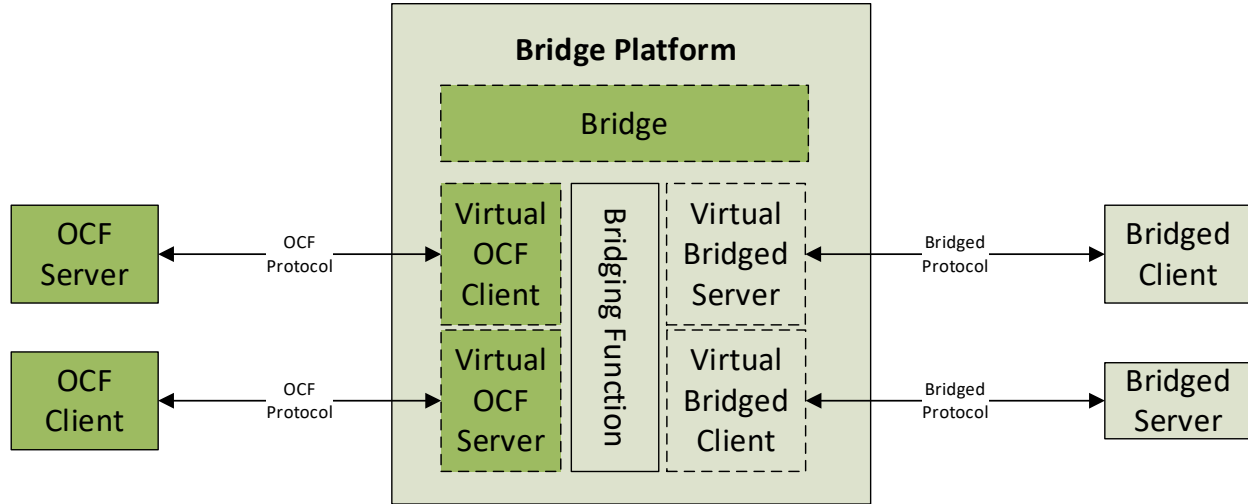
406 Strings that are to be taken literally are enclosed in "double quotes".

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

408 **5 Bridge Platform**

409 **5.1 Introduction**

410 This clause describes the functionality of a Bridge Platform; such a device is illustrated in Figure 1.



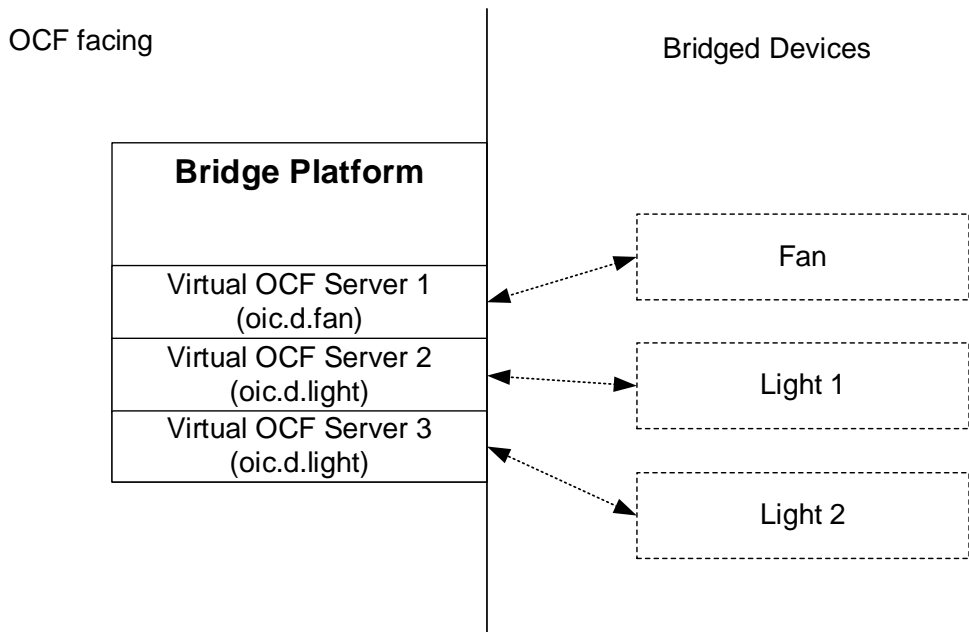
411

412

Figure 1 – Bridge Platform components

413 A Bridge Platform enables the representation of one or more Bridged Devices as Virtual OCF
414 Devices (VODs) on the network and/or enables the representation of one or more OCF Devices as
415 Virtual OCF Devices using another protocol on the network. The Bridged Devices themselves are
416 out of the scope of this document. The only difference between a native OCF Device and a VOD
417 from the perspective of an OCF Client is the inclusion of "oic.d.virtual" in the "rt" of "/oic/d" of the
418 VOD.

419 A Bridge Platform exposes a Bridge Device which is an OCF Device with a Device Type of
420 "oic.d.bridge". This provides to an OCF Client an explicit indication that the discovered Device is
421 performing a bridging function. This is useful for several reasons; 1) when establishing a home
422 network, the Client can determine that the bridge is reachable and functional when no bridged
423 devices are present, 2) allows for specific actions to be performed on the bridge considering the
424 known functionality a bridge supports, 3) allows for explicit discovery of all devices that are serving
425 a bridging function which benefits trouble shooting and maintenance actions on behalf of a user.
426 When such a device is discovered the exposed Resources on the OCF Bridge Device describe
427 other devices. For example, as shown in Figure 2.



428

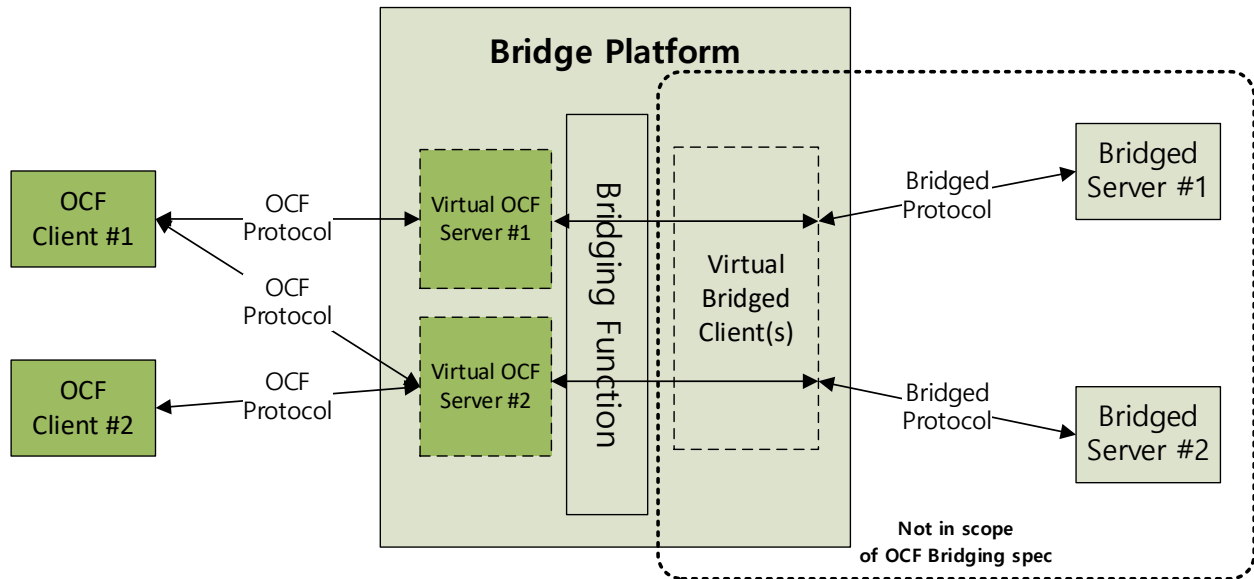
429 **Figure 2 – Schematic overview of a Bridge Platform bridging non-OCF devices**

430 It is expected that the Bridge Platform creates a set of devices during the start-up of the Bridge
 431 Platform, these being the Bridge and any known VODs. The exposed set of VODs can change as
 432 Bridged Devices are added or removed from the bridge. The adding and removing of Bridged
 433 Devices is implementation dependent.

434 **5.2 Symmetric vs. asymmetric bridging**

435 There are two kinds of bridging: Symmetric, Asymmetric. In symmetric bridging, a bridge device
 436 exposes OCF server(s) to another ecosystem and exposes other ecosystem’s server(s) to OCF. In
 437 asymmetric bridging, a bridge device exposes OCF server(s) to another ecosystem or exposes
 438 another ecosystem’s server(s) to OCF, but not both. The former case is called an Asymmetric
 439 Server Bridge (see Figure 3), the latter case is called an Asymmetric Client Bridge (see Figure 4)

440



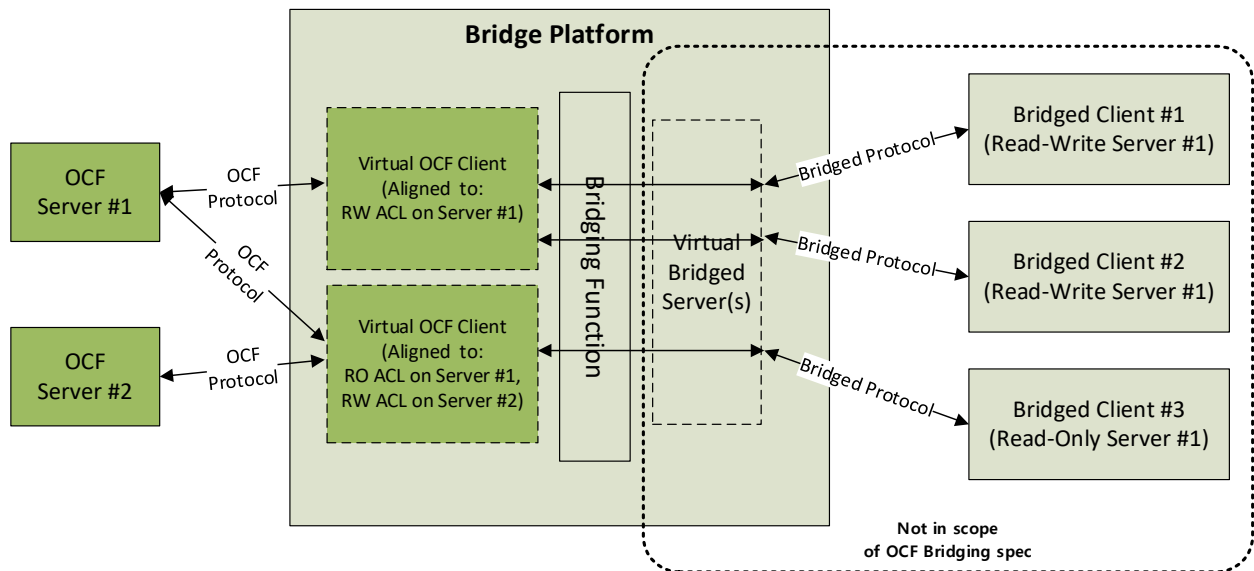
441

442

Figure 3 – Asymmetric server bridge

443 In Figure 3 each Bridged Server is exposed as a Virtual OCF Server to OCF side. These Virtual
 444 OCF Servers are same as normal OCF Servers except that they have additional rt value
 445 ("oic.d.virtual") for "/oic/d". The details of the Virtual Bridged Client are not in scope of this
 446 document.

447



448

449

Figure 4 – Asymmetric client bridge

450 Figure 4 shows that each access to the OCF Server is modelled as a Virtual OCF Client. Those
 451 accesses can be aggregated if their target OCF servers and access permissions are same,
 452 therefore a Virtual OCF Client can tackle multiple Bridged Clients.

453 **5.3 General requirements**

454 **5.3.1 For Asymmetric Bridging**

455 A VOD shall have a Device Type that contains "oic.d.virtual". This allows Bridge Platforms to
456 determine if a device is already being translated when multiple Bridge Platforms are present.

457 Each Bridged Server shall be exposed as a separate Virtual OCF Device, with its own OCF
458 Endpoint, and set of mandatory Resources (as defined in ISO/IEC 30118-1:2018 and ISO/IEC
459 30118-2:2018). Discovery of a VOD is the same as for an ordinary OCF Device; that is the VOD
460 shall respond to multicast discovery requests. This allows platform-specific, device-specific, and
461 resource-specific fields to all be preserved across translation.

462 The Bridge Introspection Device Data (IDD) provides information for the Resources exposed by the
463 Bridge only. Each VOD shall expose an instance of "oic.wk.introspection" which provides a URL to
464 an IDD for the specific VOD.

465 **5.3.2 For Symmetric Bridging**

466 In addition to the requirements mentioned in 5.3.1, Symmetric Bridging shall satisfy following
467 requirements.

468 The Bridge Platform shall check the protocol-independent UUID ("piid" in OCF) of each device and
469 shall not advertise back into a Bridged Protocol a device originally seen via that Bridged Protocol.
470 The Bridge Platform shall stop translating any Bridged Protocol device exposed in OCF via another
471 Bridge Platform if the Bridge Platform sees the device via the Bridged Protocol. Similarly, the Bridge
472 Platform shall not advertise an OCF Device back into OCF, and the Bridge Platform shall stop
473 translating any OCF device exposed in the Bridged Protocol via another Bridge Platform if the
474 Bridge Platform sees the device via OCF. These require that the Bridge Platform can determine
475 when a device is already being translated. A VOD shall be indicated on the OCF Security Domain
476 with a Device Type of "oic.d.virtual". How a Bridge Platform determines if a device is already being
477 translated on a non-OCF Security Domain is described in the protocol-specific clauses (e.g. clause
478 6).

479 The Bridge Platform shall detect duplicate VODs (with the same protocol-independent UUID)
480 present in a network and shall not create more than one corresponding virtual device as it translates
481 those duplicate devices into another network.

482 **5.4 VOD List**

483 For maintenance purposes, the Bridge maintains a list of VODs. This list includes only VODs
484 created by the Bridge. A single instance of the Resource Type that defines the VOD list (see clause
485 10.4) shall be exposed by the Bridge. Please refer to ISO/IEC 30118-2:2018 for detailed operational
486 requirements for the VOD list.

487 **5.5 Resource discovery**

488 A Bridge Platform shall detect devices that arrive and leave the Bridged network or the OCF
489 Security Domain. Where there is no pre-existing mechanism to reliably detect the arrival and
490 departure of devices on a network, a Bridge Platform shall periodically poll the network to detect
491 arrival and departure of devices, for example using COAP multicast discovery (a multicast
492 RETRIEVE of "/oic/res") in the case of the OCF Security Domain. Bridge Platform implementations
493 are encouraged to use a poll interval of 30 seconds plus or minus a random delay of a few seconds.

494 An Bridge Platform and any exposed VODs shall each respond to network discovery commands.
495 The response to a RETRIEVE on "/oic/res" shall only include the devices that match the RETRIEVE
496 request.

497 The resource reference determined from each Link exposed by "/oic/res" on the Bridge or on a
498 VOD shall be unique. The Bridge and the VODs shall meet the requirements defined in ISO/IEC
499 30118-1:2018 for population of the Properties and Link parameters in "/oic/res".

500 For example, if a Bridge exposes VODs for the fan and lights shown in Figure 2, and an OCF Client
501 performs a discovery request with a content format of "application/vnd.ocf+cbor", there will be four
502 discrete responses, one for the Bridge, one for the virtual fan Device, and two for the virtual light
503 Devices. Note that what is returned is not in the JSON format but in a suitable encoding as defined
504 in ISO/IEC 30118-1:2018.

```
505 Response from the Bridge:
506 [
507   {
508     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
509     "href": "/oic/res",
510     "rel": "self",
511     "rt": ["oic.wk.res"],
512     "if": ["oic.if.ll", "oic.if.baseline"],
513     "p": {"bm": 3},
514     "eps": [{"ep": "coap://[2001:db8:a::b1d4]:5555"},
515             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
516   },
517   {
518     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
519     "href": "/oic/d",
520     "rt": ["oic.wk.d", "oic.d.bridge"],
521     "if": ["oic.if.r", "oic.if.baseline"],
522     "p": {"bm": 3},
523     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
524   },
525   {
526     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
527     "href": "/oic/p",
528     "rt": ["oic.wk.p"],
529     "if": ["oic.if.r", "oic.if.baseline"],
530     "p": {"bm": 3},
531     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
532   },
533   {
534     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
535     "href": "/oic/sec/doxm",
536     "rt": ["oic.r.doxm"],
537     "if": ["oic.if.baseline"],
538     "p": {"bm": 1},
539     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
540   },
541   {
542     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
543     "href": "/oic/sec/pstat",
544     "rt": ["oic.r.pstat"],
545     "if": ["oic.if.baseline"],
546     "p": {"bm": 1},
547     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
548   },
549   {
550     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
551     "href": "/oic/sec/cred",
552     "rt": ["oic.r.cred"],
553     "if": ["oic.if.baseline"],
554     "p": {"bm": 1},
555     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
```

```

556     },
557     {
558         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
559         "href": "/oic/sec/acl2",
560         "rt": ["oic.r.acl2"],
561         "if": ["oic.if.baseline"],
562         "p": {"bm": 1},
563         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
564     },
565     {
566         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
567         "href": "/myIntrospection",
568         "rt": ["oic.wk.introspection"],
569         "if": ["oic.if.r", "oic.if.baseline"],
570         "p": {"bm": 3},
571         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
572     },
573     {
574         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
575         "href": "/myVodlist",
576         "rt": ["oic.r.vodlist"],
577         "if": ["oic.if.r", "oic.if.baseline"],
578         "p": {"bm": 3},
579         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
580     }
581 ]
582
583 Response from the Fan VOD:
584 [
585     {
586         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
587         "href": "/oic/res",
588         "rt": ["oic.wk.res"],
589         "if": ["oic.if.ll", "oic.if.baseline"],
590         "p": {"bm": 3},
591         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
592     },
593     {
594         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
595         "href": "/oic/d",
596         "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
597         "if": ["oic.if.r", "oic.if.baseline"],
598         "p": {"bm": 3},
599         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
600     },
601     {
602         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
603         "href": "/oic/p",
604         "rt": ["oic.wk.p"],
605         "if": ["oic.if.r", "oic.if.baseline"],
606         "p": {"bm": 3},
607         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
608     },
609     {
610         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
611         "href": "/myFan",
612         "rt": ["oic.r.switch.binary"],
613         "if": ["oic.if.a", "oic.if.baseline"],
614         "p": {"bm": 3},
615         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
616     },
617     {

```

```

618     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
619     "href": "/oic/sec/doxm",
620     "rt": ["oic.r.doxm"],
621     "if": ["oic.if.baseline"],
622     "p": {"bm": 1},
623     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
624 },
625 {
626     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
627     "href": "/oic/sec/pstat",
628     "rt": ["oic.r.pstat"],
629     "if": ["oic.if.baseline"],
630     "p": {"bm": 1},
631     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
632 },
633 {
634     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
635     "href": "/oic/sec/cred",
636     "rt": ["oic.r.cred"],
637     "if": ["oic.if.baseline"],
638     "p": {"bm": 1},
639     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
640 },
641 {
642     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
643     "href": "/oic/sec/acl2",
644     "rt": ["oic.r.acl2"],
645     "if": ["oic.if.baseline"],
646     "p": {"bm": 1},
647     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
648 },
649 {
650     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
651     "href": "/myFanIntrospection",
652     "rt": ["oic.wk.introspection"],
653     "if": ["oic.if.r", "oic.if.baseline"],
654     "p": {"bm": 3},
655     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
656 }
657 ]
658
659 Response from the first Light VOD:
660 [
661 {
662     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
663     "href": "/oic/res",
664     "rt": ["oic.wk.res"],
665     "if": ["oic.if.ll", "oic.if.baseline"],
666     "p": {"bm": 3},
667     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
668 },
669 {
670     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
671     "href": "/oic/d",
672     "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
673     "if": ["oic.if.r", "oic.if.baseline"],
674     "p": {"bm": 3},
675     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
676 },
677 {
678     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
679     "href": "/oic/p",

```

```

680     "rt": ["oic.wk.p"],
681     "if": ["oic.if.r", "oic.if.baseline"],
682     "p": {"bm": 3},
683     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
684 },
685 {
686     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
687     "href": "/myLight",
688     "rt": ["oic.r.switch.binary"],
689     "if": ["oic.if.a", "oic.if.baseline"],
690     "p": {"bm": 3},
691     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
692 },
693 {
694     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
695     "href": "/oic/sec/doxm",
696     "rt": ["oic.r.doxm"],
697     "if": ["oic.if.baseline"],
698     "p": {"bm": 1},
699     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
700 },
701 {
702     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
703     "href": "/oic/sec/pstat",
704     "rt": ["oic.r.pstat"],
705     "if": ["oic.if.baseline"],
706     "p": {"bm": 1},
707     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
708 },
709 {
710     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
711     "href": "/oic/sec/cred",
712     "rt": ["oic.r.cred"],
713     "if": ["oic.if.baseline"],
714     "p": {"bm": 1},
715     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
716 },
717 {
718     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
719     "href": "/oic/sec/acl2",
720     "rt": ["oic.r.acl2"],
721     "if": ["oic.if.baseline"],
722     "p": {"bm": 1},
723     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
724 },
725 {
726     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
727     "href": "/myLightIntrospection",
728     "rt": ["oic.wk.introspection"],
729     "if": ["oic.if.r", "oic.if.baseline"],
730     "p": {"bm": 3},
731     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
732 }
733 ]
734
735 Response from the second Light VOD:
736 [
737 {
738     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
739     "href": "/oic/res",
740     "rt": ["oic.wk.res"],
741     "if": ["oic.if.ll", "oic.if.baseline"],

```

```

742     "p": {"bm": 3},
743     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
744 },
745 {
746     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
747     "href": "/oic/d",
748     "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
749     "if": ["oic.if.r", "oic.if.baseline"],
750     "p": {"bm": 3},
751     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
752 },
753 {
754     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
755     "href": "/oic/p",
756     "rt": ["oic.wk.p"],
757     "if": ["oic.if.r", "oic.if.baseline"],
758     "p": {"bm": 3},
759     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
760 },
761 {
762     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
763     "href": "/myLight",
764     "rt": ["oic.r.switch.binary"],
765     "if": ["oic.if.a", "oic.if.baseline"],
766     "p": {"bm": 3},
767     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
768 },
769 {
770     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
771     "href": "/oic/sec/doxm",
772     "rt": ["oic.r.doxm"],
773     "if": ["oic.if.baseline"],
774     "p": {"bm": 1},
775     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
776 },
777 {
778     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
779     "href": "/oic/sec/pstat",
780     "rt": ["oic.r.pstat"],
781     "if": ["oic.if.baseline"],
782     "p": {"bm": 1},
783     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
784 },
785 {
786     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
787     "href": "/oic/sec/cred",
788     "rt": ["oic.r.cred"],
789     "if": ["oic.if.baseline"],
790     "p": {"bm": 1},
791     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
792 },
793 {
794     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
795     "href": "/oic/sec/acl2",
796     "rt": ["oic.r.acl2"],
797     "if": ["oic.if.baseline"],
798     "p": {"bm": 1},
799     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
800 },
801 {
802     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
803     "href": "/myLightIntrospection",

```

```

804     "rt": ["oic.wk.introspection"],
805     "if": ["oic.if.r", "oic.if.baseline"],
806     "p": {"bm": 3},
807     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
808   }
809 ]

```

810 **Figure 5 – /oic/res example responses**

811 **5.6 "Deep translation" vs. "on-the-fly"**

812 When translating a service between a Bridged Protocol (e.g., AllJoyn) and OCF protocols, there
813 are two possible types of translation. Bridge Platforms are expected to dedicate most of their logic
814 to "deep translation" types of communication, in which data models used with the Bridged Protocol
815 are mapped to the equivalent OCF Resource Types and vice-versa, in such a way that a compliant
816 OCF Client or Bridged Client would be able to interact with the service without realising that a
817 translation was made.

818 "Deep translation" is out of the scope of this document, as the procedure far exceeds mapping of
819 types. For example, clients on one side of a Bridge Platform may decide to represent an intensity
820 as an 8-bit value between 0 and 255, whereas the devices on the other may have chosen to
821 represent that as a floating-point number between 0.0 and 1.0. It's also possible that the procedure
822 may require storing state in the Bridge Platform. Either way, the programming of such translation
823 will require dedicated effort and study of the mechanisms on both sides.

824 The other type of translation, the "on-the-fly" or "one-to-one" translation, requires no prior
825 knowledge of the device-specific schema in question on the part of the Bridge Platform. The burden
826 is, instead, on one of the other participants in the communication, usually the client application.
827 That stems from the fact that "on-the-fly" translation always produces Bridged Resource Types and
828 OCF Resource Types as vendor extensions.

829 For AllJoyn, deep translation is specified in ISO/IEC 30118-6:2018, and on-the-fly translation is
830 covered in clause 7.2 of this document.

831 **5.7 Security**

832 Please refer to ISO/IEC 30118-2:2018 for security specific requirements as they pertain to a Bridge
833 Platform. These security requirements include both universal requirements applicable to all Bridged
834 Protocols, and additional security requirements specific to each Bridged Protocol.

835 **6 AllJoyn translation**

836 **6.1 Operational scenarios**

837 The overall goals are to:

- 838 1) make Bridged Servers appear to OCF clients as if they were native OCF servers, and
- 839 2) make OCF servers appear to Bridged Clients as if they were native non-OCF servers.

840 **6.2 Requirements specific to an AllJoyn Bridging Function**

841 **6.2.1 Introduction**

842 The Bridge Platform shall be an AllJoyn Router Node. (This is a requirement so that users can
843 expect that a certified Bridge will be able to talk to any AllJoyn device, without the user having to
844 buy some other device.)

845 The requirements in clause 6.2 apply when using algorithmic translation, and by default apply to
846 deep translation unless the relevant clause for such deep translation specifies otherwise.

847 **6.2.2 Use of introspection**

848 Whenever possible, the translation code should make use of metadata available that indicates what
849 the sender and recipient of the message in question are expecting. For example, devices that are
850 AllJoyn Certified are required to carry the introspection data for each object and interface they
851 expose. When the metadata is available, Bridging Functions should convert the incoming payload
852 to exactly the format expected by the recipient and should use information when translating replies
853 to form a more useful message.

854 For example, for an AllJoyn specific Bridging Function, the expected interaction list is presented in
855 Table 1.

856 **Table 1 – AllJoyn Bridging Function Interaction List**

Message Type	Sender	Receiver	Metadata
Request	AllJoyn 16.10	OCF 1.0	Available
Request	OCF 1.0	AllJoyn 16.10	Available
Response	AllJoyn 16.10	OCF 1.0	Available
Response	OCF 1.0	AllJoyn 16.10	Available

857 **6.2.3 Stability and loss of data**

858 Round-tripping through the translation process specified in this document is not expected to
859 reproduce the same original message. The process is, however, designed not to lose data or
860 precision in messages, though it should be noted that both OCF and AllJoyn payload formats allow
861 for future extensions not considered in this document.

862 However, a third round of translation should produce the same identical message as was previously
863 produced, provided the same information is available. That is, in the chain shown in Figure 6,
864 payloads 2 and 4 as well as 3 and 5 should be identical.

865
866 **Figure 6 – Payload Chain.**

867 **6.2.4 Exposing AllJoyn producer devices to OCF clients**

868 **6.2.4.1 Virtual OCF Devices and Resources**

869 As specified in ISO/IEC 30118-2:2018 the value of the "di" property of OCF Devices (including
870 VODs) shall be established as part of Onboarding of that VOD.

871 Each AllJoyn object shall be mapped to one or more Virtual OCF Resources. If all AllJoyn interfaces
872 can be translated to resource types on the same resource, there should be a single Virtual OCF
873 Resource, and the path component of the URI of the Virtual OCF Resource shall be the AllJoyn
874 object path, where each "_h" in the AllJoyn object path is transformed to "-" (hyphen), each "_d" in
875 the AllJoyn object path is transformed to "." (dot), each "_t" in the AllJoyn object path is transformed
876 to "~" (tilde), and each "_u" in the AllJoyn object path is transformed to "_" (underscore). Otherwise,
877 a Resource with that path shall exist with a Resource Type of ["oic.wk.col", "oic.r.alljoynobject"]
878 which is a Collection of links, where "oic.r.alljoynobject" is defined in clause 10.2 and the items in
879 the collection are the Resources with the translated Resource Types.

880 The value of the "piid" property of "/oic/d" for each VOD shall be the value of the OCF-defined
881 AllJoyn field "org.openconnectivity.piid" in the AllJoyn About Announce signal, if that field exists,
882 else it shall be calculated by the Bridging Function as follows:

883 – If the AllJoyn device supports security, the value of the "piid" property value shall be the peer
884 GUID.

885 – If the AllJoyn device does not support security but the device is being bridged anyway (see
886 10.2), the "piid" property value shall be derived from the Deviceld and Appld properties (in the
887 About data), by concatenating the Deviceld value (not including any null termination) and the
888 Appld bytes and using the result as the "name" to be used in the algorithm specified in
889 IETF RFC 4122 clause 4.3, with SHA-1 as the hash algorithm, and 8f0e4e90-79e5-11e6-bdf4-
890 0800200c9a66 as the name space ID. (This is to address the problem of being able to de-
891 duplicate AllJoyn devices exposed via separate OCF Bridge Devices.)

892 A Bridging Function implementation is encouraged to listen for AllJoyn About Announce signals
893 matching any AllJoyn interface name. It can maintain a cache of information it received from these
894 signals, and use the cache to quickly handle "/oic/res" queries from OCF Clients (without having to
895 wait for Announce signals while handling the queries).

896 A Bridging Function implementation is encouraged to listen for other signals (including
897 EmitsChangedSignal of properties) only when there is a client subscribed to a corresponding
898 resource on a Virtual AllJoyn Device.

899 There are multiple types of AllJoyn interfaces, which shall be handled as follows.

900 1) If the AllJoyn interface is in a well-defined set (defined in ISO/IEC 30118-6:2018 or 6.2.4.2) of
901 interfaces where standard forms exist on both the AllJoyn and OCF sides, the Bridging Function
902 shall either:

903 a) follow the specification for translating that interface specially, or

904 b) not translate the AllJoyn interface.

905 2) If the AllJoyn interface is not in the well-defined set, the Bridging Function shall either:

906 a) not translate the AllJoyn interface, or

907 b) algorithmically map the AllJoyn interface as specified in 6.3 to custom/vendor-defined
908 Resource Types by converting the AllJoyn interface name to OCF resource type name(s).

909 An AllJoyn interface name shall be converted to a Device Type or a set of one or more OCF
910 Resource Types as follows:

911 1) If the AllJoyn interface has any members, append a suffix ".<seeBelow>" where <seeBelow> is
912 described in this clause.

913 2) For each upper-case letter present in the entire string, replace it with a hyphen followed by the
914 lower-case version of that letter (e.g., convert "A" to "-a").

915 3) If an underscore appears followed by a (lower-case) letter or a hyphen, for each such
916 occurrence, replace the underscore with two hyphens (e.g., convert "_a" to "--a", "_-a" to "---
917 a").

918 4) For each underscore remaining, replace it with a hyphen (e.g., convert "_1" to "-1").

919 5) Prepend the "x." prefix.

920 Some examples are shown in Table 2. The first three are normal AllJoyn names converted to
921 unusual OCF names. The last three are unusual AllJoyn names converted (perhaps back) to normal
922 OCF names. ("xn--" is a normal domain name prefix for the Punycode-encoded form of an
923 Internationalized Domain Name, and hence can appear in a normal vendor-specific OCF name.)

Table 2 – AllJoyn to OCF Name Examples

From AllJoyn name	To OCF name
example.Widget	x.example.-widget
example.my__widget	x.example.my----widget
example.My_Widget	x.example.-my---widget
xn_p1ai.example	x.xn--p1ai.example
xn__90ae.example	x.xn--90ae.example
example.myName_1	x.example.my-name-1

925 Each AllJoyn interface that has members and is using algorithmic mapping shall be mapped to one
926 or more Resource Types as follows:

- 927 – AllJoyn Properties with the same EmitsChangedSignal value are mapped to the same Resource
928 Type where the value of the <seeBelow> label is the value of EmitsChangedSignal. AllJoyn
929 Properties with EmitsChangedSignal values of "const" or "false", are mapped to Resources that
930 are not Observable, whereas AllJoyn Properties with EmitsChangedSignal values of "true" or
931 "invalidates" result in Resources that are Observable. The Version property in an AllJoyn
932 interface is always considered to have an EmitsChangedSignal value of "const", even if not
933 specified in introspection XML. The name of each property on the Resource Type shall be
934 "<ResourceType>.<AllJoynPropertyName>", where each "_d" in the <AllJoynPropertyName> is
935 transformed to "." (dot), and each "_h" in the <AllJoynPropertyName> is transformed to "-"
936 (hyphen).
- 937 – Resource Types mapping AllJoyn Properties with access "readwrite" shall support the "oic.if.rw"
938 OCF Interface. Resource Types mapping AllJoyn Properties with access "read" shall support
939 the "oic.if.r" OCF Interface. Resource Types supporting both the "oic.if.rw" and "oic.if.r" OCF
940 Interfaces shall choose "oic.if.r" as the default Interface.
- 941 – Each AllJoyn Method is mapped to a separate Resource Type, where the value of the
942 <seeBelow> label is the AllJoyn Method name. The Resource Type shall support the "oic.if.rw"
943 OCF Interface. Each argument of the AllJoyn Method shall be mapped to a separate Property
944 on the Resource Type, where the name of that Property is prefixed with
945 "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the AllJoyn
946 introspection xml, in order to help get uniqueness across all Resource Types on the same
947 Resource. Therefore, when the AllJoyn argument name is not specified, the name of that
948 property is "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in
949 the AllJoyn introspection XML. In addition, that Resource Type has an extra
950 "<ResourceType>validity" property that indicates whether the rest of the properties have valid
951 values. When the values are sent as part of an UPDATE response, the validity property is true,
952 and any other properties have valid values. In a RETRIEVE (GET or equivalent in the relevant
953 transport binding) response, the validity property is false, and any other properties can have
954 meaningless values. If the validity property appears in an UPDATE request, its value shall be
955 true (a value of false shall result in an error response).
- 956 – Each AllJoyn Signal (whether sessionless, sessioncast, or unicast) is mapped to a separate
957 Resource Type on an Observable Resource, where the value of the <seeBelow> label is the
958 AllJoyn Signal name. The Resource Type shall support the "oic.if.r" OCF Interface. Each
959 argument of the AllJoyn Signal is mapped to a separate Property on the Resource Type, where
960 the name of that Property is prefixed with "<ResourceType>arg<#>", where <#> is the 0-indexed
961 position of the argument in the AllJoyn introspection xml, in order to help get uniqueness across
962 all Resource Types on the same Resource. Therefore, when the AllJoyn argument name is not
963 specified, the name of that property is "<ResourceType>arg<#>", where <#> is the 0-indexed
964 position of the argument in the AllJoyn introspection XML. In addition, that Resource Type has
965 an extra "<ResourceType>validity" property that indicates whether the rest of the properties

966 have valid values. When the values are sent as part of a NOTIFY response, the validity property
 967 is true, and any other properties have valid values. In a RETRIEVE (GET or equivalent in the
 968 relevant transport binding) response, the validity property is false, and any other properties
 969 returned can have meaningless values. This is because in AllJoyn, the signals are
 970 instantaneous events, and the values are not necessarily meaningful beyond the lifetime of that
 971 message. Note that AllJoyn does have a TTL field that allows store-and-forward signals, but
 972 such support is not required in OCF 1.0. We expect that in the future, the TTL may be used to
 973 allow valid values in response to a RETRIEVE that is within the TTL.

974 When an algorithmic mapping is used, AllJoyn data types shall be mapped to OCF property types
 975 according to 6.3.

976 If an AllJoyn operation fails, the Bridging Function shall send an appropriate OCF error response
 977 to the OCF client. If an AllJoyn error name is available and does not contain the
 978 "org.openconnectivity.Error.Code" prefix, it shall construct an appropriate OCF error message (e.g.,
 979 diagnostic payload if using CoAP) from the AllJoyn error name and AllJoyn error message (if any),
 980 using the form "<error name>: <error message>", with the <error name> taken from the AllJoyn
 981 error name field and the <error message> taken from the AllJoyn error message, and the CoAP
 982 error code set to an appropriate value (if CoAP is used). If an AllJoyn error name is available and
 983 contains the "org.openconnectivity.Error.Code" prefix, the OCF error message (e.g., diagnostic
 984 payload if using CoAP) should be taken from the AllJoyn error message (if any), and the CoAP
 985 error code (if CoAP is used) set to a value derived as follows; remove the
 986 "org.openconnectivity.Error.Code" prefix, and if the resulting error name is of the form "<#>" where
 987 <#> is an error code without a decimal (e.g., "404"), the CoAP error code shall be the error code
 988 indicated by the "<#>". Example: "org.openconnectivity.Error.Code404" becomes "404", which shall
 989 result in an error 4.04 for a CoAP transport.

990 **6.2.4.2 Exposing an AllJoyn producer application as a Virtual OCF Server**

991 Table 3 shows how OCF Device properties, as specified in Table 27 in ISO/IEC 30118-1:2018 shall
 992 be derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn
 993 Configuration Interface Specification.

994 If the AllJoyn About or Config data field has a mapping rule defined (as in Table 3, Table 4, Table 5,
 995 and Table 6), the field name shall be translated based on that mapping rule; else if the AllJoyn
 996 About or Config data field has a fully qualified name (with a <domain> prefix (such as
 997 "com.example", "org.alljoyn"), the field name shall be translated based on the rules specified in
 998 6.2.4 for mapping AllJoyn fields; else, the field shall not be translated as it may be incorrect (error)
 999 or it has no valid mapping (such as daemonRealm and passCode).

1000 **Table 3 – oic.wk.d resource type definition**

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
(Device) Name	n	Human friendly name For example, "Bob's Thermostat"	Y	AppName (no exact equivalent exists)	Application name assigned by the app manufacturer (developer or the OEM).	Y
Spec Version	icv	Spec version of ISO/IEC 30118-1:2018 this device is implemented to, the syntax is "core.major.minor"]	Y	(none)	Bridge Platform should return its own value	N

Device ID	di	Unique identifier for Device. This value shall be as defined in ISO/IEC 30118-2:2018 for DeviceID.	Y	(none)	Use as defined in ISO/IEC 30118-2:2018	N
Protocol-Independent ID	piid	Unique identifier for OCF Device (UUID)	Y	<p>org.openconnectivity.piid if it exists, else "Peer GUID" (not in About, but exposed by protocol) if authenticated, else Hash(DeviceId,AppId) where the Hash is done by concatenating the Device Id (not including any null terminator) and the AppId and using the algorithm in IETF RFC 4122 clause 4.3, with SHA-1.</p> <p>This means that the value of di may change if the resource is read both before and after authentication, in order to mitigate privacy concerns discussed in RFC 6973.</p>	<p>Peer GUID: The peer GUID is the only persistent identity for a peer. Peer GUIDs are used by the authentication mechanisms to uniquely identify a remote application instance. The peer GUID for a remote peer is only available if the remote peer has been authenticated.</p> <p>DeviceId: Device identifier set by platform-specific means.</p> <p>AppId: A 128-bit globally unique identifier for the application. The AppId shall be a universally unique identifier as specified in IETF RFC 4122.</p>	<p>Per GUID: conditionally Y</p> <p>DeviceId: Y</p> <p>AppId: Y</p>
Data Model Version	dmv	Spec version(s) of the vertical specifications this device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor". <vertical> is the name of the vertical (i.e. sh for Smart Home)	Y	<p>Comma separated list of the Version property values of each interface listed in the objectDescription argument of the Announce signal of About. In addition to the mandatory values specified in ISO/IEC 30118-1:2018, additional values are formatted as "x.<interface name>.<Version property value>".</p>	<p>This document assumes that the value of the Version property is the same as the value of the "org.gtk.GDBus.Since" annotation of the interface in the AllJoyn introspection XML, and therefore the value of the Version property may be determined through introspection alone.</p> <p>Note that AllJoyn specifies that the default value is 1 if the "org.gtk.GDBus.Since" annotation is absent.</p>	<p>N, but required by IRB for all standard interfaces, and absence can be used to imply a constant (e.g., 0)</p>

Localized Descriptions	ld	Detailed description of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the device description in the indicated language.	N	Description	Detailed description expressed in language tags as in RFC 5646.	Y
Software Version	sv	Version of the device software.	N	SoftwareVersion	Software version of the app.	Y
Manufacturer Name	dmn	Name of manufacturer of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the manufacturer name in the indicated language.	N	Manufacturer	The manufacturer's name of the app.	Y
Model Number	dmno	Model number as designated by manufacturer.	N	ModelNumber	The app model number.	Y

1001

1002 In addition, any additional vendor-defined fields in the AllJoyn About data shall be mapped to
 1003 vendor-defined properties in the OCF Device resource "/oic/d" (which implements the "oic.wk.d"
 1004 resource type), with a property name formed by prepending "x." to the AllJoyn field name.

1005 Table 4 shows how OCF Device Configuration properties, as specified in Table 22 in ISO/IEC
 1006 30118-1:2018 shall be derived:

1007

Table 4 – oic.wk.con resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
(Device) Name	n	Human friendly name For example, "Bob's Thermostat"	Y	AppName (no exact equivalent exists)	Application name assigned by the app manufacturer (developer or the OEM).	Y
Location	loc	Provides location information where available.	N	org.openconnectivity.loc (if it exists, else property shall be absent)		N

Location Name	locn	Human friendly name for location For example, "Living Room".	N	org.openconnectivity.locn (if it exists, else property shall be absent)		N
Currency	c	Indicates the currency that is used for any monetary transactions	N	org.openconnectivity.c (if it exists, else property shall be absent)		N
Region	r	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").	N	org.openconnectivity.r (if it exists, else property shall be absent)		N
Localized Names	ln	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.	N	AppName	Application name assigned by the app manufacturer (developer or the OEM).	Y
Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this language unless the property	N	DefaultLanguage	The default language supported by the device. Specified as an IETF language tag listed in RFC 5646.	Y

		specifies otherwise.				
--	--	----------------------	--	--	--	--

1008
1009 In addition, any additional vendor-defined fields in the AllJoyn Configuration data shall be mapped
1010 to vendor-defined properties in the OCF Configuration resource (which implements the "oic.wk.con"
1011 resource type and optionally the "oic.wk.con.p" resource type), with a property name formed by
1012 prepending "x." to the AllJoyn field name.

1013 Table 5 shows how OCF Platform properties, as specified in Table 28 in ISO/IEC 30118-1:2018
1014 shall be derived, typically from fields specified in the AllJoyn About Interface Specification and
1015 AllJoyn Configuration Interface Specification.

1016 **Table 5 – oic.wk.p resource type definition**

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
Platform ID	pi	Unique identifier for the physical platform (UUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.	Y	DeviceId if it is a UUID, else generate a name-based UUID from the DeviceId using the DeviceId value (not including any null termination) as the "name" to be used in the algorithm specified in IETF RFC 4122 clause 4.3, with SHA-1 as the hash algorithm, and 8f0e4e90-79e5-11e6-bdf4-0800200c9a66 as the name space ID.	Name of the device set by platform-specific means (such as Linux and Android).	Y
Manufacturer Name	mnmn	Name of manufacturer (not to exceed 16 characters)	Y	Manufacturer (in DefaultLanguage, truncated to 16 characters)	The manufacturer's name of the app.	Y
Manufacturer Details Link (URL)	mnml	URL to manufacturer (not to exceed 32 characters)	N	org.openconnectivity.mnml (if it exists, else property shall be absent)		N
Model Number	mnmo	Model number as designated by manufacturer	N	ModelNumber	The app model number.	Y
Date of Manufacture	mndt	Manufacturing date of device	N	DateOfManufacture	Date of manufacture using format YYYY-MM-DD (known as XML DateTime format).	N

Platform Version	mnpv	Version of platform – string (defined by manufacturer)	N	org.openconnectivity.mnpv (if it exists, else property shall be absent)		N
OS Version	mnos	Version of platform resident OS – string (defined by manufacturer)	N	org.openconnectivity.mnos (if it exists, else property shall be absent)		N
Hardware Version	mnhw	Version of platform hardware	N	HardwareVersion	Hardware version of the device on which the app is running.	N
Firmware version	mnfv	Version of device firmware	N	org.openconnectivity.mnfv (if it exists, else property shall be absent)		N
Support URL	mnsi	URL that points to support information from manufacturer	N	SupportUrl	Support URL (populated by the manufacturer)	N
SystemTime	st	Reference time for the device	N	org.openconnectivity.st (if it exists, else property shall be absent)		N
Vendor ID	vid	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.	N	DeviceId	Name of the device set by platform-specific means (such as Linux and Android).	Y

1017 Table 6 shows how OCF Platform Configuration properties, as specified in Table 23 in the ISO/IEC
1018 30118-1:2018 shall be derived:

1019 **Table 6 – oic.wk.con.p resource type definition**

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From AJ Field name	AJ Description	AJ Mandatory
Platform Names	Mnpr	Platform Identifier	N	DeviceName	Name of the device set by platform-specific means (such as Linux and Android).	Device name assigned by the user. The device name appears on the UI as the friendly name of the device.

1020
1021 In addition, the "oic.wk.mnt" properties Factory_Reset ("fr") and Reboot ("rb") shall be mapped to
1022 AllJoyn Configuration methods FactoryReset and Restart, respectively.

1023 **6.2.5 Exposing OCF resources to AllJoyn consumer applications**

1024 **6.2.5.1 Use of AllJoyn Producer Application**

1025 Unless specified otherwise, each OCF resource shall be mapped to a separate AllJoyn object.

1026 Each OCF Server shall be exposed as a separate AllJoyn producer application, with its own About
1027 data. This allows platform-specific, device-specific, and resource-specific fields to all be preserved
1028 across translation. However, this requires that AllJoyn Claiming of such producer applications be
1029 solved in a way that does not require user interaction, but this is left as an implementation issue.

1030 The AllJoyn producer application shall implement the "oic.d.virtual" AllJoyn interface. This allows
1031 Bridge Platforms to determine if a device is already being translated when multiple Bridge Platforms
1032 are present. The "oic.d.virtual" interface is defined as follows:

```
1033 <interface name="oic.d.virtual"/>
```

1034 The implementation may choose to implement this interface by the AllJoyn object at path "/oic/d".

1035 The AllJoyn peer ID shall be the OCF device ID ("di").

1036 Unless specified otherwise, the AllJoyn object path shall be the OCF URI path, where each "-"
1037 (hyphen) in the OCF URI path is transformed to "_h", each "." (dot) in the OCF URI path is
1038 transformed to "_d", each "~" (tilde) in the OCF URI path is transformed to "_t", and each "_"
1039 (underscore) in the OCF URI path is transformed to "_u".

1040 The AllJoyn About data shall be populated per Table 8.

1041 A Bridging Function implementation is encouraged to maintain a cache of OCF resources to handle
1042 the implementation of queries from the AllJoyn side, and emit an Announce Signal for each OCF
1043 Server. Specifically, the implementation could always Observe "/oic/res" changes and only Observe
1044 other resources when there is a client with a session on a Virtual AllJoyn Device.

1045 There are multiple types of resources, which shall be handled as follows.

1046 1) If the Resource Type is in a well-defined set (defined in ISO/IEC 30118-6:2018 or 6.2.5.2) of
1047 resource types where standard forms exist on both the AllJoyn and OCF sides, the Bridging
1048 Function shall either:

- 1049 a) follow the specification for translating that resource type specially, or
- 1050 b) not translate the Resource Type.

1051 2) If the Resource Type is not in the well-defined set (but is not a Device Type), the Bridging
1052 Function shall either:

- 1053 a) not translate the Resource Type, or
- 1054 b) algorithmically map the Resource Type as specified in 6.3 to a custom/vendor-defined
1055 AllJoyn interface by converting the OCF Resource Type name to an AllJoyn Interface name.

1056 An OCF Resource Type or Device Type shall be converted to an AllJoyn interface name as follows:

1057 1) Remove the "x." prefix if present

1058 2) For each occurrence of a hyphen (in order from left to right in the string):

1059 a) If the hyphen is followed by a letter, replace both characters with a single upper-case version
1060 of that letter (e.g., convert "-a" to "A").

1061 b) Else, if the hyphen is followed by another hyphen followed by either a letter or a hyphen,
1062 replace two hyphens with a single underscore (e.g., convert "--a" to "_a", "---" to "_-").

1063 c) Else, convert the hyphen to an underscore (i.e., convert "-" to "_").

1064 Some examples are shown in the Table 7. The first three are unusual OCF names converted
 1065 (perhaps back) to normal AllJoyn names. The last three are normal OCF names converted to
 1066 unusual AllJoyn names. ("xn--" is a normal domain name prefix for the Punycode-encoded form of
 1067 an Internationalized Domain Name, and hence can appear in a normal vendor-specific OCF name.)

1068 **Table 7 – Example name mapping**

From OCF name	To AllJoyn name
x.example.-widget	example.Widget
x.example.my----widget	example.my__widget
x.example.-my---widget	example.My_Widget
x.xn--p1ai.example	xn_p1ai.example
x.xn--90ae.example	xn__90ae.example
x.example.my-name-1	example.myName_1

1069 An OCF Device Type is mapped to an AllJoyn interface with no members.

1070 Unless specified otherwise, each OCF Resource Type shall be mapped to an AllJoyn interface as
 1071 follows:

- 1072 – Each OCF property is mapped to an AllJoyn property in that interface, where each "." (dot) in
 1073 the OCF property is transformed to "_d", and each "-" (hyphen) in the OCF property is
 1074 transformed to "_h".
- 1075 – The EmitsChangedSignal value for each AllJoyn property shall be set to "true" if the resource
 1076 supports NOTIFY, or "false" if it does not. (The value is never set to "const" or "invalidates"
 1077 since those concepts cannot currently be expressed in OCF.)
- 1078 – The "access" attribute for each AllJoyn property shall be "read" if the OCF property is read-only,
 1079 or "readwrite" if the OCF property is read-write.
- 1080 – If the resource supports DELETE, a Delete() method shall appear in the interface.
- 1081 – If the resource supports CREATE, a Create() method shall appear in the interface, with input
 1082 arguments of each property of the resource to create. (Such information is not available
 1083 algorithmically can be determined via introspection.) If such information is not available, a
 1084 CreateWithDefaultValues() method shall appear which takes no input arguments. In either case,
 1085 the output argument shall be an OBJECT_PATH containing the path of the created resource.
- 1086 – If the resource supports UPDATE (i.e., the "oic.if.rw" or "oic.if.a" OCF Interface) then an AllJoyn
 1087 property set operation (i.e., an org.freedesktop.DBus.Properties.Set() method call) shall be
 1088 mapped to a Partial UPDATE (e.g., POST in CoAP) with the corresponding OCF property.
- 1089 – If a Resource has a Resource Type "oic.r.alljoynobject", then instead of separately translating
 1090 each of the Resources in the collection to its own AllJoyn object, all Resources in the collection
 1091 shall instead be translated to a single AllJoyn object whose object path is the OCF URI path of
 1092 the collection.

1093 OCF property types shall be mapped to AllJoyn data types according to 6.3.

1094 If an OCF operation fails, the Bridging Function shall send an appropriate AllJoyn error response
 1095 to the AllJoyn consumer. If an error message is present in the OCF response, and the error
 1096 message (e.g., diagnostic payload if using CoAP) fits the pattern "<error name>: <error message>"
 1097 where <error name> conforms to the AllJoyn error name syntax requirements, the AllJoyn error
 1098 name and AllJoyn error message shall be extracted from the error message in the OCF response.
 1099 Otherwise, the AllJoyn error name shall be "org.openconnectivity.Error.Code<#>" where <#> is the

1100 error code (e.g., CoAP error code) in the OCF response without a decimal (e.g., "404") and the
 1101 AllJoyn error message is the error message in the OCF response.

1102 **6.2.5.2 Exposing an OCF server as a Virtual AllJoyn Producer**

1103 The object description returned in the About interface shall be formed as specified in the AllJoyn
 1104 About Interface Specification, and Table 8 shows how AllJoyn About Interface fields shall be
 1105 derived, based on properties in "oic.wk.d", "oic.wk.con", "oic.wk.p", and "oic.wk.con.p".

1106

Table 8 – AllJoyn about data fields

To AJ Field name	AJ Description	AJ Mandatory	From OCF Property title	OCF Property name	OCF Description	OCF Mandatory
Appld	A 128-bit globally unique identifier for the application. The Appld shall be a universally unique identifier as specified in RFC 4122.	Y	Device ID (no exact equivalent exists)	di	Unique identifier for OCF Device (UUID)	Y
DefaultLanguage	The default language supported by the device. Specified as an IETF language tag listed in RFC 5646.	Y	Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this language unless the property specifies otherwise. If absent, the Bridge Platform shall return a constant, e.g., empty string	N
DeviceName (per supported language)	Name of the device set by platform-specific means (such as Linux and Android).	N	Platform Names	mnpn	Friendly name of the Platform. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the platform friendly name in the indicated language.	N

					For example, [{"language": "en", "value": "Dave's Laptop"}]	
DeviceId	Device identifier set by platform-specific means.	Y	Platform ID	pi	Platform Identifier	Y
AppName (per supported language)	Application name assigned by the app manufacturer (developer or the OEM).	Y	Localized Names, if it exists, else (Device) Name	In or n	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.	N (In), Y (n)
Manufacturer (per supported language)	The manufacturer's name of the app.	Y	Manufacturer Name	dmn	Name of manufacturer of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the manufacturer name in the indicated language.	N
ModelNumber	The app model number.	Y	Model Number	dmno	Model number as designated by manufacturer	N
SupportedLanguages	List of supported languages.	Y	language fields of Localized Names	In	If In is supported, return the list of values of the language field of each array	N

					element, else return empty array	
Description (per supported language)	Detailed description expressed in language tags as in RFC 5646.	Y	Localized Descriptions	ld	Detailed description of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the device description in the indicated language.	N
DateOfManufacture	Date of manufacture using format YYYY-MM-DD (known as XML DateTime format).	N	Date of Manufacture	mndt	Manufacturing date of device	N
SoftwareVersion	Software version of the app.	Y	Software Version	sv	Software version of the device.	N
AJSoftwareVersion	Current version of the AllJoyn SDK used by the application.	Y	(none)		Bridge Platform should return its own value	
HardwareVersion	Hardware version of the device on which the app is running.	N	Hardware Version	mnhw	Version of platform hardware	N
SupportUrl	Support URL (populated by the manufacturer)	N	Support URL	mnsI	URL that points to support information from manufacturer	N
org.openconnectivity.mnml		N	Manufacturer Details Link (URL)	mnml (if it exists, else field shall be absent)	URL to manufacturer (not to exceed 32 characters)	N
org.openconnectivity.mnpv		N	Platform Version	mnpv (if it exists, else field shall be absent)	Version of platform – string (defined by manufacturer)	N
org.openconnectivity.mnos		N	OS Version	mnos (if it exists,	Version of platform resident	N

				else field shall be absent)	OS – string (defined by manufacturer)	
org.openconnectivity.mnfv		N	Firmware version	mnfv (if it exists, else field shall be absent)	Version of device firmware	N
org.openconnectivity.st		N	SystemTime	st (if it exists, else field shall be absent)	Reference time for the device	N
org.openconnectivity.piid		N	Protocol-Independent ID	piid	A unique and immutable Device identifier. A Client can detect that a single Device supports multiple communication protocols if it discovers that the Device uses a single Protocol Independent ID value for all the protocols it supports.	Y

1107

1108 The AllJoyn field "org.openconnectivity.piid" shall be announced but shall not be localized and its
1109 D-Bus type signature shall be "s". All other AllJoyn field names listed in Table 5 which have the
1110 prefix "org.openconnectivity." shall be neither announced nor localized and their D-Bus type
1111 signature shall be "s".

1112 In addition, any additional vendor-defined properties in the OCF Device resource "/oic/d" (which
1113 implements the "oic.wk.d" resource type) and the OCF Platform resource "/oic/p" (which
1114 implements the "oic.wk.p" resource type) shall be mapped to vendor-defined fields in the AllJoyn
1115 About data, with a field name formed by removing the leading "x." from the property name.

1116 Table 9 shows how AllJoyn Configuration Interface fields shall be derived, based on properties in
1117 "oic.wk.con" and "oic.wk.con.p".

1118

Table 9 – AllJoyn configuration data fields

To AJ Field name	AJ Description	AJ Mandatory	From OCF Property title	OCF Property name	OCF Description	OCF Mandatory
DefaultLanguage	Default language supported by the device.	N	Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any	N

					string property as being in this language unless the property specifies otherwise.	
DeviceName	Device name assigned by the user. The device name appears on the UI as the friendly name of the device.	N	PlatformNames	mnpn	Friendly name of the Platform. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the platform friendly name in the indicated language. For example, [{"language": "en", "value": "Dave's Laptop"}]	N
org.openconnectivity.loc		N	Location	loc (if it exists, else field shall be absent)	Provides location information where available.	N
org.openconnectivity.locn		N	Location Name	locn (if it exists, else field shall be absent)	Human friendly name for location For example, "Living Room".	N
org.openconnectivity.c		N	Currency	c (if it exists, else field shall be absent)	Indicates the currency that is used for any monetary transactions	N
org.openconnectivity.r		N	Region	r (if it exists, else field shall be absent)	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").	N

1119

1120 The AllJoyn field "org.openconnectivity.loc" shall be neither announced nor localized and its D-Bus
1121 type signature shall be "ad". All other AllJoyn field names listed in Table 5 which have the prefix
1122 "org.openconnectivity." shall be neither announced nor localized and their D-Bus type signature
1123 shall be "s".

1124 In addition, the Configuration methods FactoryReset and Restart shall be mapped to "oic.wk.mnt"
 1125 properties Factory_Reset ("fr") and Reboot ("rb"), respectively, and any additional vendor-defined
 1126 properties in the OCF Configuration resource (which implements the "oic.wk.con" resource type
 1127 and optionally the "oic.wk.con.p" resource type) shall be mapped to vendor-defined fields the
 1128 AllJoyn Configuration data, with a field name formed by removing the leading "x." from the property
 1129 name.

1130 **6.2.6 Security**

1131 For AllJoyn bridging, an OCF Onboarding Tool shall be able to block the communication of all OCF
 1132 Devices with all Bridged Devices that don't communicate securely with the Bridge, by using the
 1133 Bridge Device's "oic.r.securemode" Resource.

1134 **6.3 On-the-Fly Translation from D-Bus and OCF payloads**

1135 **6.3.1 Introduction**

1136 The "dbus1" payload format is specified in the D-Bus Specification and AllJoyn adopted the D-Bus
 1137 protocol and made it distributed over the network. The modifications done by AllJoyn to the format
 1138 are all in the header part of the packet, not in the data payload itself, which remains compatible
 1139 with "dbus1". Other variants of the protocol that have been proposed by the Linux community
 1140 ("GVariant" and "kdbus" payloads) contain slight incompatibilities and are not relevant for this
 1141 discussion.

1142 **6.3.2 Translation without aid of introspection**

1143 **6.3.2.1 Introduction**

1144 Clause 6.3.2 describes how Bridging Functions shall translate messages between the two payload
 1145 formats in the absence of introspection metadata from the actual device. This situation arises in
 1146 the when there is content not described by introspection, such as the inner payload of AllJoyn
 1147 properties of type "D-Bus VARIANT".

1148 Since introspection is not available, the Bridging Function cannot know the rich JSON sub-type,
 1149 only the underlying CBOR type and from that it can infer the JSON generic type, and hence
 1150 translation is specified in terms of those generic types.

1151 **6.3.2.2 Booleans**

1152 Boolean conversion is trivial since both sides support this type.

1153 **Table 10 – Boolean translation**

D-Bus type	JSON type
"b" – BOOLEAN	boolean (true or false)

1154

1155 **6.3.2.3 Numeric types**

1156 The translation of numeric types is lossy and that is unavoidable due to the limited expressiveness
 1157 of the JSON generic types. This can only be solved with introspection.

1158 The translation of numeric types is direction-specific.

1159 **Table 11 – Numeric type translation, D-Bus to JSON**

From D-Bus type	To JSON type
"y" - BYTE (unsigned 8-bit)	Number

"n" - UINT16 (unsigned 16-bit)	
"u" - UINT32 (unsigned 32-bit)	
"t" - UINT64 (unsigned 64-bit) ^a	
"q" - INT16 (signed 16-bit)	
"" - INT32 (signed 32-bit)	
"x" - INT64 (signed 64-bit) ^a	
"d" - DOUBLE (IEEE 754 double precision)	
<p>^a D-Bus payloads of types "t" (UINT64) and "x" (INT64) can contain values that cannot be perfectly represented in IEEE 754 double-precision floating point. The RFCs governing JSON do not forbid such numbers but caution that many implementations may not be able to deal with them. Currently, OCF transports its payload using CBOR instead of JSON, which can represent those numbers with fidelity. However, it should be noted that ISO/IEC 30118-1:2018 does not allow for integral numbers outside the range $-2^{53} \leq x \leq 2^{53}$.</p>	

1160

1161

Table 12 – Numeric type translation, JSON to D-Bus

From JSON type	To D-Bus type
number	"d" - DOUBLE ^a
<p>^a To provide the most predictable result, all translations from OCF to AllJoyn produce values of type "d" DOUBLE (IEEE 754 double precision).</p>	

1162

1163

1164 **6.3.2.4 Text strings**

1165

Table 13 – Text string translation

D-Bus type	JSON type
"s" – STRING	string

1166 Conversion between D-Bus and JSON strings is simple, as both require their content to be valid
 1167 Unicode. For example, an implementation can typically do a direct byte copy, as both protocols
 1168 specify UTF-8 as the encoding of the data, neither constrains the data to a given normalisation
 1169 format nor specify whether private-use characters or non-characters should be disallowed.

1170 Since the length of D-Bus strings is always known, it is recommended Bridging Functions not use
 1171 CBOR indeterminate text strings (first byte 0x7f).

1172 **6.3.2.5 Byte arrays**

1173 The translation of a byte array is direction-specific.

1174

Table 14 – Byte array translation

From D-Bus type	To JSON type
"ay" - ARRAY of BYTE	(base64-encoded) string

1175 The base64url encoding is specified in IETF RF 4648 clause 5.

1176 **6.3.2.6 D-Bus variants**

1177 **Table 15 – D-Bus variant translation**

D-Bus type	JSON type
"v" – VARIANT	see clause 6.3.2.6

1178
1179 D-Bus has a type called VARIANT ("v") that is a wrapper around any other D-Bus type. It's a way
1180 for the type system to perform type-erasure. JSON, on the other hand, is not type-safe, which
1181 means that all JSON values are, technically, variants. The conversion for a D-Bus variant to JSON
1182 is performed by entering that variant and encoding the type carried inside as per the rules in this
1183 document.

1184 The algorithm must be recursive, as D-Bus variants are allowed to contain variants themselves.

1185 **6.3.2.7 D-Bus object paths and signatures**

1186 The translation of D-Bus object paths and signatures is unidirectional (there is no mapping *to* them,
1187 only *from* them). This is shown in Table 16. In the reverse direction, clause 6.3.2.4 always converts
1188 to D-Bus STRING rather than OBJECT_PATH or SIGNATURE since it is assumed that "s" is the
1189 most common string type in use.

1190 **Table 16 – D-Bus object path translation**

From D-Bus type	To JSON type
"o" - OBJECT_PATH	string
"g" – SIGNATURE	

1191
1192 Both D-Bus object paths and D-Bus type signatures are US-ASCII strings with specific formation
1193 rules, found in the D-Bus Specification. They are very seldom used and are not expected to be
1194 found in resources subject to translation without the aid of introspection.

1195 **6.3.2.8 D-Bus structures**

1196 The translation of the types in Table 17 is direction-specific:

1197 **Table 17 – D-Bus structure translation**

From D-Bus type	To JSON type
"r" – STRUCT	array, length > 0

1198
1199 D-Bus structures can be interpreted as a fixed-length array containing a pre-determined list of types
1200 for each member. This is how such a structure is mapped to JSON: as an array of heterogeneous
1201 content, which are the exact members of the D-Bus structure, in the order in which they appear in
1202 the structure.

1203 **6.3.2.9 Arrays**

1204 The translation of the types in Table 18 is bidirectional:

1205

Table 18 – Byte array translation

D-Bus type	JSON type
"ay" - ARRAY of BYTE	(base64-encoded) string – see 6.3.2.5
"ae" - ARRAY of DICT_ENTRY	object – see 6.3.2.10

1206

1207

1208 The translation of the types in Table 19 is direction-specific:

1209

Table 19 – Other array translation

From D-Bus type	To JSON type
"a" – ARRAY of anything else not specified	array

1210

1211 Aside from arrays of bytes and arrays of dictionary entries, which are mapped to JSON strings and
1212 objects respectively, arrays in JSON cannot be constrained to a single type (i.e., heterogeneous
1213 arrays). For that reason, strictly speaking all D-Bus arrays excepting arrays of bytes and arrays of
1214 dictionary entries must first be converted to arrays of variant "av" and then that array can be
1215 converted to JSON. See Table 20.

1216

Table 20 – JSON array translation

From JSON type	Condition	To D-Bus type
array	length=0	"av" – ARRAY of VARIANT
array	length>0, all elements of same type	"a" – ARRAY
array	length>0, elements of different types	"r" – STRUCT

1217 Conversion of D-Bus arrays of variants uses the conversion of variants as specified, which simply
1218 eliminates the distinction between a variant containing a given value and that value outside a
1219 variant. In other words, the elements of a D-Bus array are extracted and sent as elements of the
1220 JSON array, as per the other rules of this document.

1221 **6.3.2.10 Dictionaries / Objects**

1222 The choice of "dictionary of STRING to VARIANT" is made because that is the most common type
1223 of dictionary found in payloads and is an almost perfect superset of all possible dictionaries in D-
1224 Bus anyway. Moreover, it can represent JSON Objects with fidelity, which is the representation that
1225 OCF uses in its data models, which in turn means those D-Bus dictionaries will be able to carry
1226 with fidelity any OCF JSON Object in current use. See Table 21

1227

Table 21 – D-Bus dictionary translation

D-Bus type	JSON type
"a{sv}" - dictionary of STRING to VARIANT	object

1228 D-Bus dictionaries that are not mapping string to variant are first converted to those constraints
1229 and then encoded in CBOR.

1230 **6.3.2.11 Non-translatable types**

1231 The types in Table 22 are not translatable, and the Bridging Function should drop the incoming
1232 message. None of the types in Table 22 are in current use by either AllJoyn or OCF 1.0 devices,
1233 so the inability to translate them should not be a problem.

1234

Table 22 – Non-translation types

Type Scope	Type Name	Description
D-Bus	"h"	UNIX_FD (Unix File Descriptor)
JSON	Null	
JSON	undefined	Not officially valid JSON, but some implementations permit it

1235

1236 **6.3.2.12 Examples**

1237 Table 23 and Table 24 provide some translation examples.

Table 23 – D-Bus to JSON translation examples

Source D-Bus	JSON Result
BOOLEAN(FALSE)	false
BOOLEAN(TRUE)	true
VARIANT(BOOLEAN(FALSE))	false
VARIANT(BOOLEAN(TRUE))	true
BYTE(0)	0.0
BYTE(255)	255.0
INT16(0)	0.0
INT16(-1)	-1.0
INT16(-32768)	-32768.0
UINT16(0)	0.0
UINT16(65535)	65535.0
INT32(0)	0.0
INT32(-2147483648)	-2147483648.0
INT32(2147483647)	2147483647.0
UINT32(0)	0.0
UINT32(4294967295)	4294967295.0
INT64(0)	0.0
INT64(-1)	-1.0
UINT64(18446744073709551615)	18446744073709551615.0 ⁽¹⁾
DOUBLE(0.0)	0.0
DOUBLE(0.5)	0.5
STRING("")	""
STRING("Hello")	"Hello"
ARRAY<BYTE>()	""
ARRAY<BYTE>(0x48, 0x65, 0x6c, 0x6c, 0x6f)	"SGVsbG8"
OBJECT_PATH("/")	"/"
SIGNATURE()	""
SIGNATURE("s")	"s"
VARIANT(INT32(0))	0
VARIANT(VARIANT(INT32(0)))	0
VARIANT(STRING("Hello"))	"Hello"

1239

1240

Table 24 – JSON to D-Bus translation examples

Source JSON	D-Bus Result
false	BOOLEAN(false)
true	BOOLEAN(true)
0	DOUBLE(0.0)
-1	DOUBLE(-1.0)
-2147483648	DOUBLE(-2147483648.0)
2147483647	DOUBLE(2147483647.0)
2147483648	DOUBLE(2147483648.0)
-2147483649	DOUBLE(-2147483649.0)
9223372036854775808 ⁽¹⁾	DOUBLE(9223372036854775808.0)
0.0	DOUBLE(0.0)
0.5	DOUBLE(0.5)
0.0f	DOUBLE(0.0)
0.5f	DOUBLE(0.5)
""	STRING("")
"Hello"	STRING("Hello")
[]	ARRAY<VARIANT>()
[1]	ARRAY<DOUBLE>(DOUBLE(1.0))
[1, 2147483648, false, "Hello"]	STRUCT<DOUBLE, DOUBLE, BOOLEAN, STRING>(DOUBLE(1.0), DOUBLE(2147483648.0), BOOLEAN(false), STRING("Hello"))
{}	map<STRING, VARIANT>()
{1: 1}	map<STRING, VARIANT>("1" → VARIANT(DOUBLE(1.0)))
{"1": 1}	map<STRING, VARIANT>("1" → VARIANT(DOUBLE(1.0)))
{ "rep": { "state": false, "power": 1.0, "name": "My Light" } }	map<STRING, VARIANT>({STRING("rep"), VARIANT(map<STRING, VARIANT>({STRING("state") → VARIANT(BOOLEAN(FALSE))}, {STRING("power") → VARIANT(DOUBLE(1.0))}, {STRING("name") → VARIANT(STRING("My Light"))}))})

1241 NOTE This value cannot be represented with IEEE754 double-precision floating point without loss of information. It is
1242 also outside the currently-allowed range of integrals in OCF.

1243 6.3.3 Translation with aid of introspection

1244 6.3.3.1 Introduction to Introspection Metadata

1245 When introspection is available, the Bridging Function can use the extra metadata provided by the
1246 side offering the service to expose a higher-quality reply to the other side. This chapter details
1247 modifications to the translation described in the previous chapter when the metadata is found.

1248 Introspection metadata can be used for both translating requests to services and replies from those
1249 services. When used to translate requests, the introspection is "constraining", since the Bridging
1250 Function must conform exactly to what that service expects. When used to translate replies, the
1251 introspection is "relaxing", but may be used to inform the receiver what other possible values may
1252 be encountered in the future.

1253 Note that OCF introspection uses JSON types, media attributes, and format attributes, not CBOR
1254 encoding. The actual encoding of each JSON type is discussed in clause 12.4 of ISO/IEC 30118-
1255 1:2018 , JSON format attribute values are as defined in JSON Schema Validation, and JSON media
1256 attribute values are as defined in JSON Hyper-Schema.

1257 **6.3.3.2 Translation of the introspection itself**

1258 Note that both OCF 1.0 and AllJoyn require all services exposed to include introspection metadata,
1259 which means the Bridging Function will need to translate the introspection information on-the-fly
1260 for each OCF resource or AllJoyn producer it finds. The Bridging Function shall preserve as much
1261 of the original information as can be represented in the translated format. This includes both the
1262 information used in machine interactions and the information used in user interactions, such as
1263 description and documentation text.

1264 **6.3.3.3 Variability of introspection data**

1265 Introspection data is not a constant and the Bridging Function may find, upon discovering further
1266 services, that the D-Bus interface or OCF Resource Type it had previously encountered is different
1267 than previously seen. The Bridging Function needs to take care about how the destination side will
1268 react to a change in introspection.

1269 D-Bus interfaces used by AllJoyn services may be updated to newer versions, which means a given
1270 type of service may be offered by two distinct versions of the same interface. Updates to
1271 standardised interfaces must follow strict guidelines established by the AllSeen Interface Review
1272 Board, mapping each version to a different OCF Resource Type should be possible without much
1273 difficulty. However, there's no guarantee that vendor-specific extensions follow those requirements.
1274 Indeed, there's nothing preventing two revisions of a product to contain completely incompatible
1275 interfaces that have the same name and version number.

1276 On the opposite direction, the rules are much laxer. Since OCF specifies optional properties to its
1277 Resource Types, a simple monotonically-increasing version number like AllJoyn consumer
1278 applications expect is not possible.

1279 However, it should be noted that services created by the Bridging Function by "on-the-fly"
1280 translation will only be accessed by generic client applications. Dedicated applications will only use
1281 "deep binding" translation.

1282 **6.3.3.4 Numeric types**

1283 For numeric values, all D-Bus and JSON numeric types are treated equally as source and may all
1284 be translated into any of the other side's types. When translating a request to a service, the Bridging
1285 Function need only verify whether there would be loss of information when translating from source
1286 to destination. For example, when translating the number 1.5 to either a JSON integer or to one of
1287 the D-Bus integral types, there would be loss of information, in which case the Bridging Function
1288 should refuse the incoming message. Similarly, the value 1,234,567 does not fit the range of a D-
1289 Bus byte, 16-bit signed or unsigned integer.

1290 When translating the reply from the service, the Bridging Function shall use the following rules.

1291 Table 25 indicates how to translate from a JSON type to the corresponding D-Bus type, where the
1292 first matching row shall be used. If the JSON schema does not indicate the minimum value of a
1293 JSON integer, 0 is the default. If the JSON schema does not indicate the maximum value of a JSON

1294 integer, $2^{32} - 1$ is the default. The resulting AllJoyn introspection XML shall contain
 1295 "org.alljoyn.Bus.Type.Min" and "org.alljoyn.Bus.Type.Max" annotations whenever the minimum or
 1296 maximum, respectively, of the JSON value is different from the natural minimum or maximum of
 1297 the D-Bus type.

1298

Table 25 – JSON type to D-Bus type translation

From JSON type	Condition	To D-Bus Type
integer	minimum ≥ 0 AND maximum $< 2^8$	"y" (BYTE)
	minimum ≥ 0 AND maximum $< 2^{16}$	"q" (UINT16)
	minimum $\geq -2^{15}$ AND maximum $< 2^{15}$	"n" (INT16)
	minimum ≥ 0 AND maximum $< 2^{32}$	"u" (UINT32)
	minimum $\geq -2^{31}$ AND maximum $< 2^{31}$	"i" (INT32)
	minimum ≥ 0	"t" (UINT64)
		"x" (INT64)
Number		"d" (DOUBLE)
String	pattern = <code>^0 ([1-9][0-9]{0,19})\$</code>	"t" (UINT64)
	pattern = <code>^0 (-?[1-9][0-9]{0,18})\$</code>	"x" (INT64)

1299 Table 26 indicates how to translate from a D-Bus type to the corresponding JSON type.

1300

Table 26 – D-Bus type to JSON type translation

From D-Bus type	To JSON type	Note
"y" (BYTE)	integer	"minimum" and "maximum" in the JSON schema shall be set to the value of the "org.alljoyn.Bus.Type.Min" and "org.alljoyn.Bus.Type.Max" (respectively) annotations if present, or to the min and max values of the D-Bus type's range if such annotations are absent.
"n" (UINT16)		
"q" (INT16)		
"u" (UINT32)		
"i" (INT32)		
"t" (UINT64)	integer if org.alljoyn.Bus.Type.Max $\leq 2^{53}$, else string with JSON pattern attribute <code>^0 ([1-9][0-9]{0,19})\$</code> .	IETF RFC 7159 clause 6 explains that higher JSON integers are not interoperable.
"x" (INT64)	integer (if org.alljoyn.Bus.Type.Min $\geq -2^{53}$ AND org.alljoyn.Bus.Type.Max $\leq 2^{53}$), else string with JSON pattern attribute <code>^0 (-?[1-9][0-9]{0,18})\$</code> .	IETF RFC 7159 clause 6 explains that other JSON integers are not interoperable.
"d" (double)	number	

1301

1302

1303 **6.3.3.5 Text string and byte arrays**

1304 There's no difference in the translation of text strings and byte arrays compared to clause 6.3.2.
 1305 Clause 6.3.3 simply lists the JSON equivalent types for the generated OCF introspection. See
 1306 Table 27.

1307 **Table 27 – Text string translation**

D-Bus Type	JSON type	JSON media attribute, binaryEncoding property
"s" – STRING	string	(none)
"ay" - ARRAY of BYTE	string	base64

1308 In addition, the mapping of the JSON Types in Table 28 is direction-specific:

1309 **Table 28 – JSON UUID string translation**

From JSON type	Condition	To D-Bus Type
string	pattern = "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}\$"	"ay" – ARRAY of BYTE

1310 JSON strings with any other format value (e.g., date-time, uri, etc.) or pattern value not shown in
 1311 Table 28 shall be treated the same as if the format and pattern attributes were absent, by simply
 1312 mapping the value to a D-Bus string.

1313 **6.3.3.6 D-Bus Variants**

1314 If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus
 1315 VARIANT, the Bridging Function should create such a variant and encode the incoming value as
 1316 the variant's payload as per the rules in the rest of this document. See Table 29.

1317 **Table 29 – D-Bus variant translation**

D-Bus Type	JSON Type
"v" – VARIANT	see clause 6.3.3.6

1318 **6.3.3.7 D-Bus Object Paths and Signatures**

1319 If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus Object
 1320 Path or D-Bus Signature, the Bridging Function should perform a validity check in the incoming
 1321 CBOR Text String. If the incoming data fails to pass this check, the message should be rejected.
 1322 See Table 30.

1323 **Table 30 – D-Bus object path translation**

From D-Bus Type	To JSON Type
"o" – OBJECT_PATH	string
"g" – SIGNATURE	

1324 **6.3.3.8 D-Bus structures**

1325 D-Bus structure members are described in the introspection XML with the
 1326 "org.alljoyn.Bus.Struct.*StructureName*.Field.*fieldName*.Type" annotation. The Bridging Function
 1327 shall use the AJSoftwareVersion field of the About data obtained from a bridged AllJoyn producer
 1328 as follows. When the version of AllJoyn implemented on the Bridged Device is v16.10.00 or greater
 1329 and the member annotations are present, the Bridging Function shall use a JSON object to

1330 represent a structure, mapping each member to the entry with that name. The Bridging Function
 1331 needs to be aware that the incoming CBOR payload may have changed the order of the fields,
 1332 when compared to the D-Bus structure. When the version of AllJoyn implemented on the Bridged
 1333 Device is less than v16.10.00, the Bridging Function shall follow the rule for translating D-Bus
 1334 structures without the aid of introspection data.

1335 **6.3.3.9 Arrays and dictionaries**

1336 If the introspection of the AllJoyn interface indicates that the array is neither an ARRAY of BYTE
 1337 ("ay") nor an ARRAY of VARIANT ("av") or that the dictionary is not mapping STRING to VARIANT
 1338 ("a{sv}"), the Bridging Function shall apply the constraining or relaxing rules specified in other
 1339 clauses.

1340 Similarly, if the OCF introspection indicates a homogeneous array type, the information about the
 1341 array's element type should be used as the D-Bus array type instead of VARIANT ("v").

1342 **6.3.3.10 Other JSON format attribute values**

1343 The JSON format attribute may include other custom attribute types. They are not known at this
 1344 time, but it is expected that those types be handled by their type and representation alone.

1345 **6.3.3.11 Examples**

1346 Table 31 and Table 32 provide examples using introspection.

1347 **Table 31 – Mapping from AllJoyn using introspection**

AllJoyn Source	AllJoyn Introspection Notes	Translated JSON Payload	OCF Introspection Notes
UINT32 (0)		0	JSON schema should indicate: "type": "integer", "minimum": 0, "maximum": 4294967295
INT64 (0)		0	Since no Min/Max annotations exist in AllJoyn, JSON schema should indicate: "type": "string", "pattern": "^0 (-?[1-9][0-9]{0,18})\$"
UINT64 (0)		"0"	Since no Max annotation exists in AllJoyn, JSON schema should indicate: "type": "string", "pattern": "^0 ([1-9][0-9]{0,19})\$"
STRING("Hello")		"Hello"	JSON schema should indicate: "type": "string"
OBJECT_PATH("/")		"/"	JSON schema should indicate: "type": "string"
SIGNATURE("g")		"g"	JSON schema should indicate: "type": "string"
ARRAY<BYTE>(0x48, 0x65, 0x6c, 0x6c, 0x6f)		"SGVsbG8"	JSON schema should indicate: "type": "string", "media binaryEncoding": "base64"
VARIANT(<i>anything</i>)		?	JSON schema should indicate:

			"type": ["boolean", "object", "array", "number", "string", "integer"]
ARRAY<INT32>()		[]	JSON schema should indicate: "type": "array", "items": { "type": "integer" }
ARRAY<INT64>()		[]	JSON schema should indicate: "type": "array", "items": { "type": "string", "pattern": "^0 ([1-9][0-9]{0,18})\$" }
STRUCT< INT32, INT32>(0, 1)	AllJoyn introspection specifies the argument with the annotation: <pre><struct name="Point"> <field name="x" type="i" /> <field name="y" type="i" /> </struct></pre>	{"x": 0, "y": 1}	JSON schema should indicate: "type": "object", "properties": { "x": { "type": "integer" }, "y": { "type": "integer" } }

1348

1349

Table 32 – Mapping from CBOR using introspection

CBOR Payload	OCF Introspection Notes	Translated AllJoyn	AllJoyn Introspection Notes
0	"type": "integer"	INT32(0)	
0	"type": "integer", "minimum": -240, "maximum": 240	INT64(0)	org.alljoyn.Bus.Type.Min = -240 org.alljoyn.Bus.Type.Max = 240
0	"type": "integer", "minimum": 0, "maximum": 248	UINT64(0)	org.alljoyn.Bus.Type.Max = 248
0.0	"type": "number"	DOUBLE(0.0)	
[1]	JSON schema indicates: "type": "array", "items": { "type": "integer", "minimum": 0, "maximum": 246 }	ARRAY<UINT64>(1)	org.alljoyn.Bus.Type.Max = 246

1350

1351 7 oneM2M Translation

1352 7.1 Operational Scenarios

1353 The purpose of the oneM2M Bridge Platform is to enable access by the oneM2M ecosystem to
1354 select OCF Servers. This is accomplished by creating Virtual OCF Clients to represent the
1355 necessary access levels to the OCF servers that are exposed to the oneM2M ecosystem. The

1356 oneM2M Bridge Platform then exposes native oneM2M entities that map to those Virtual OCF
1357 Clients.

1358 The oneM2M Bridge Platform is an Asymmetric Client Bridge.

1359 The mapping between the OCF data models and the oneM2M data models is specified in OCF
1360 Resource to oneM2M Module Class Mapping. Programmatic (i.e. On-the-fly) data model translation
1361 is not supported.

1362 **7.2 Enabling oneM2M Application access to OCF Servers**

1363 Each level of oneM2M application access for OCF servers is modelled as a Virtual OCF Client. In
1364 this way, oneM2M application access can be appropriately restricted and enforced by the OCF
1365 security capabilities.

1366 **7.3 Enabling OCF Client access to oneM2M Devices**

1367 This capability is not supported.

1368 **7.4 On-the-fly Translation**

1369 All devices and resources have been aligned between the OCF and oneM2M ecosystems, so on-
1370 the-fly translation is not required.

1371 If new OCF devices are not reflected into the oneM2M ecosystem by updates to the oneM2M
1372 specifications, the Bridge Platform will not provide a successful translation of those devices.

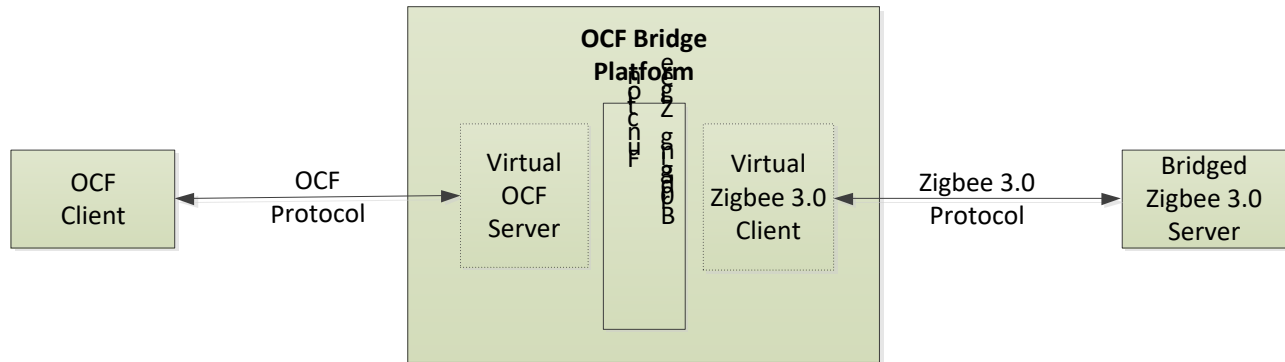
1373 **8 Zigbee Translation**

1374 **8.1 Operational Scenarios**

1375 The overall goal is to make Bridged Zigbee 3.0 Servers appear to OCF Clients as if they were
1376 native OCF Servers in the local network or cloud environment

1377 The mapping between the OCF data models and Zigbee Clusters is specified in OCF Resource to
1378 Zigbee Cluster Mapping. Programmatic (i.e. On-the-fly) data model translation is not supported.

1379 Figure 7 shows an overview of a Zigbee 3.0 Bridge Platform and its general topology. It exposes
1380 Zigbee 3.0 Servers to OCF Clients. Each Bridged Zigbee 3.0 Server is represented as a Virtual
1381 OCF Server. The Zigbee 3.0 Bridging Function supports Asymmetric bridging. The scope of this
1382 document is the asymmetric bridging to expose the Zigbee Server to OCF. The asymmetric bridging
1383 to expose an OCF Server to a Zigbee Client is out of scope.



1385

1386

Figure 7 – OCF Zigbee Bridge Platform and Components

1387

8.2 Requirements specific to Zigbee Bridging Function

1388

8.2.1 Requirements specific to Zigbee

1389

1390

1391

This document refers to Zigbee 3.0 or higher. Zigbee 3.0 is built on Zigbee Pro 2015 or newer, which enhances the IEEE 802.15.4 standard by adding a mesh network and security layers along with an application framework. Low power support is not the scope of this document.

1392

1393

1394

1395

1396

An OCF Zigbee Bridging Function shall act as a Zigbee Coordinator in network layer. A Zigbee Coordinator is responsible for initiating and maintaining the devices on the network. An OCF Zigbee Bridge Platform will act as Zigbee Client towards the Zigbee 3.0 Devices in the application layer. Users can expect that a certified OCF Bridge Platform will be able to talk to Zigbee 3.0 Devices, without the user having to buy some other device.

1397

Please see clause 5.3 for general requirements.

1398

8.2.2 Exposing Zigbee 3.0 Servers to OCF Clients

1399

1400

1401

The nature of how Zigbee Devices are structured may be different than how an OCF Device is structured. The mapping of the structure of a Zigbee device on an OCF Device is given by Table 33.

1402

1403

1404

A Zigbee Server cluster may map to one or more OCF Resources. If a specific Zigbee Server cluster has specific commands, one or more OCF Resources corresponding to the specific command attributes may be additionally needed.

1405

1406

1407

1408

1409

A Zigbee Attribute of a Zigbee Server cluster typically maps to an OCF Resource Property. However, in some special cases, multiple attributes are mapped to a single OCF Resource Property e.g., "CurrentX" and "CurrentY" of the Zigbee color control cluster map to the "csc" Property in the "oic.r.colour.csc" (Colour Space Coordinates) Resource because of the difference in the data types, i.e., "csc" is an array, but "CurrentX" and "CurrentY" map to a number.

1410

Table 34 is a mapping example of this rule

1411

Table 33 – Translation Rule between Zigbee and OCF Data Models

From Zigbee	mapping count	To OCF	mapping count
Zigbee Device	1	OCF Device	1
Zigbee Cluster	1	OCF Resource	n
Zigbee Attribute	1	OCF Resource Property	1

1412

Table 34 – Zigbee to OCF Mapping Example (Color Temperature Light)

From Zigbee		To OCF	
Zigbee 3.0 Device	0x010c (Color Temperature Light)	OCF Device	oic.d.light (Light)
Zigbee Server Cluster	0x0006 (On/Off)	OCF Resource(s)	oic.r.switch.binary (Binary Switch)
	0x0300 (Color Control Cluster)		oic.r.colour.hs (Colour Hue and Saturation)
			oic.r.colour.csc (Colour Space Coordinates)
			oic.r.colour.colourtemperature (Colour Temperature)
Zigbee Attribute	0x0000 (OnOff of On/Off Cluster)	OCF Resource Property	value (of Binary Switch Resource)
	0x0003 (CurrentX of Color Control Cluster)		csc (of Colour Space Coordinates)
	0x0004 (CurrentY of Color Control Cluster)		

1413 If a Zigbee 3.0 Device, Zigbee Server Cluster, Zigbee Attribute are enlisted in the well-defined set
 1414 (Please see OCF Resource to Zigbee Cluster Mapping), the Bridging Function shall follow the
 1415 specification for translating it to an OCF Device, OCF Resource, or OCF Resource Property
 1416 (i.e., "deep translation").

1417 A Zigbee 3.0 Server Device maps to a single OCF Device Type. The OCF Device Type is provided
 1418 by using the Device ID of the Zigbee 3.0 Server Device (The Device ID is allocated by the Zigbee
 1419 Alliance and has the same meaning of the OCF Device Type). The Zigbee 3.0 Bridging Function
 1420 has a table which includes the mapping information between the Zigbee Device ID and the OCF
 1421 Device Type. Based on the table, the Zigbee 3.0 Bridging Function finds the OCF Device Type
 1422 according to the Zigbee Device ID.

1423 A Zigbee Device includes one or more Zigbee Server Clusters. If a Zigbee Cluster maps to multiple
 1424 OCF Resources, the Zigbee Cluster may be translated as a Resource with a Collection Resource
 1425 Type. The resource mapping between Zigbee Server Cluster and OCF Resources is defined in

1426 OCF Resource to Zigbee Cluster Mapping for deep translation. The Zigbee 3.0 Bridging Function
 1427 has a table which includes the mapping information between the identifier of Zigbee Cluster and
 1428 OCF Resource Type(s). The Zigbee 3.0 Bridging Function obtains the list of cluster identifiers after
 1429 the Virtual Zigbee 3.0 Client and Zigbee 3.0 Server Device are bound. Based on the table, the
 1430 Zigbee 3.0 Bridging Function finds the OCF Resource Type(s) according to the identifier of Zigbee
 1431 Cluster.

1432 Since a Bridging Function knows all relationships between OCF Resources and Zigbee Server
 1433 Clusters, the path component of URI can be free to choose. Maintaining relationship information
 1434 and URI definition is implementation specific.

1435 If a Zigbee operation fails, the Bridging Function send an appropriate OCF error response to the
 1436 OCF Client. it construct an appropriate OCF error message (e.g., diagnostic payload if using CoAP)
 1437 from the Zigbee enumerated status value and Zigbee enumerated status (if any), using the form
 1438 "<error name>: <error message>", with the <error name> taken from the Zigbee Status Code field
 1439 and the <error message> taken from the Zigbee enumerated status, and the error code for the OCF
 1440 network set to an appropriate value.

1441 **8.2.3 Translation for well-defined set**

1442 If a Zigbee 3.0 Device, Zigbee Server Cluster, Zigbee Attribute are enlisted in the well-defined set
 1443 (Please see OCF Resource to Zigbee Cluster Mapping), the Bridging Function shall follow the
 1444 specification for translating it to an OCF Device, OCF Resource, or OCF Resource Property
 1445 (i.e., "deep translation"). Table 35 is the list of Zigbee 3.0 devices and mandatory Zigbee Server
 1446 Clusters with corresponding OCF devices and mandatory OCF Resources. Optional OCF
 1447 Resources mapped with the specific Zigbee Server Clusters are enlisted in the well-defined set
 1448 (Please see OCF Resource to Zigbee Cluster Mapping).

1449 **Table 35 – Zigbee 3.0 Device & Cluster – OCF Device & Resource mapping**

Zigbee 3.0 Device Name (Device ID)	Zigbee 3.0 Mandatory Cluster	OCF Mandatory Resource Type	OCF Device Type ("rt")	OCF Device Name
On/off light (0x0100)	On/off	oic.r.switch.binary,	oic.d.light	Light
Color Temperature Light (0x010c)	On/off, Level Control, Color Control	oic.r.switch.binary,	oic.d.light	Light
Extended Color Light (0x010d)	On/off, Level Control, Color Control	oic.r.switch.binary,	oic.d.light	Light
Dimmable Light (0x0101)	On/off, Level Control	oic.r.switch.binary,	oic.d.light	Light
Color Dimmable Light (0x0102)	On/off Level Control, Color Control	oic.r.switch.binary,	oic.d.light	Light

Zigbee 3.0 Device Name (Device ID)	Zigbee 3.0 Mandatory Cluster	OCF Mandatory Resource Type	OCF Device Type ("rt")	OCF Device Name
Temperature Sensor (0x0302)	Temperature Measurement	oic.r.temperature	oic.d.sensor	Generic Sensor
Thermostat (0x0301)	Thermostat	oic.r.temperature(2)	oic.d.thermostat	Thermostat
Window Covering Device (0x0202)	Window Covering	oic.r.openlevel	oic.d.blind	Blind
Smart Plug (0x0051)	On/off, Metering	oic.r.switch.binary,	oic.d.smartplug	Smart Plug
Mains Power Outlet (0x0009)	On/off	oic.r.switch.binary,	oic.d.smartplug	Smart Plug
On/off output (0x0002)	On/off	oic.r.switch.binary,	oic.d.smartplug	Smart Plug
IAS Zone (0x0402)	IAS Zone	oic.r.ias.zone	oic.d.sensor	Generic Sensor
Occupancy Sensor (0x0107)	Occupancy Sensing	oic.r.sensor.presence	oic.d.sensor	Generic Sensor

1450 **8.2.4 Exposing a Zigbee 3.0 Server as a Virtual OCF Server**

1451 Table 36 shows how OCF Platform properties, as specified in ISO/IEC 30118-1:2018, shall be
 1452 derived, typically from fields of Descriptor specified in Zigbee.

1453 **Table 36 – "oic.wk.p" Resource Type mapping**

To Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
Platform ID	pi	Unique identifier for the physical platform (UIUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.	Y	(none)	Bridging Function should return a randomly-generated UUID (Please see section 4.4 of IETF RFC 4122 for randomly-generated UUID)	

To Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
Manufacturer Name	mnmn	Name of manufacturer (not to exceed 16 characters)	Y	Manufacturer name (in DefaultLanguage, truncated to 16 characters)	Name of the manufacturer as a ZigBee character string Defined in Basic Cluster	Y
Manufacturer Details Link (URL)	mnml	URL to manufacturer (not to exceed 32 characters)	N	(none)	(none)	N
Model Number	mnmo	Model number as designated by manufacturer	N	Model Identifier	Model number (or other identifier) assigned by the manufacturer as a ZigBee character string Defined in Basic Cluster	Y
Date of Manufacture	mndt	Manufacturing date of device	N	DateCode	Date of manufacturer of the device in international date notation according to ISO 8601, i.e., YYYYMMDD, Defined in Basic Cluster	N
Platform Version	mpv	Version of platform – string (defined by manufacturer)	N	(none)	(none)	N
OS Version	mnos	Version of platform resident OS – string (defined by manufacturer)	N	(none)	(none)	N
Hardware Version	mnhw	Version of platform hardware	N	HWVersion	Version number of the hardware of the device. Defined in Basic Cluster	N
Firmware version	mnfv	Version of device firmware	N	(none)	(none)	N
Support link	mnsi	URI that points to support information from manufacturer	N	ProductURL	Link to a web page containing specific product information Defined in Basic Cluster	N
SystemTime	st	Reference time for the device	N	(none)	(none)	N

To Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
Vendor ID	vid	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.	N	(none)	(none)	N

1454 Table 37 shows how OCF Device Properties, as specified in Table 20 in ISO/IEC 30118-1:2018,
1455 shall be derived, typically from fields of Descriptor or Attributes of Basic cluster specified in Zigbee
1456 and Zigbee Cluster Library, respectively.

1457 As specified in ISO/IEC 30118-2:2018, the value of the “di” Property of OCF Devices (including
1458 Virtual OCF Devices) shall be established as part of Onboarding of that Virtual OCF Device.

1459 **Table 37 – "oic.wk.d" Resource Type mapping**

To Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
(Device) Name	n	Human friendly name For example, "Bob's Thermostat"	Y	User description if it exists, else Model Name if it exists, else translate Application Device Identifier (=Device ID) to Human friendly name by using Application Device Identifier value/description table	User description : Information that allows the user to identify the device using a user-friendly character string, such as "Bedroom TV" Defined in User Descriptor Model Name : character string representing the name of the manufacturer's model of the device Defined in Complex Descriptor Application Device Identifier: device description supported on this endpoint Cluster Defined in Simple Descriptor	User description: N Model Name: N Application Device Identifier: Y
Spec Version	icv	Spec version of the core specification this device is implemented to, The syntax is "core.major.minor"]	Y	(none)	Spec version of the core specification that the Bridging Platform implements should return its own value	

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
Device ID	di	Unique identifier for Device. This value shall be as defined in OCF Security Specification for DeviceID.	Y	(none)	Use as defined in the ISO/IEC 30118-2:2018	
Protocol-Independent ID	piid	Unique identifier for OCF Device (UUID)	Y	(none)	Bridging Function should return a randomly-generated UUID (Please see section 4.4 of IETF RFC 4122 for randomly-generated UUID)	
Data Model Version	dmv	Spec version(s) of the vertical specifications this device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor"]. <vertical> is the name of the vertical (i.e. sh for Smart Home)	Y	(none)	Bridging Function should return its own value.	
Localized Descriptions	ld	Detailed description of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the device description in the indicated language.	N	(none)	Zigbee provides Language and Character Set field only which specifies the language and character set used by the character strings by using ISO 639-1 language code	

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
Software Version	sv	Version of the device software.	N	ApplicationVersion	Version number of the application software contained in the device. Defined in Basic Cluster	Y
Manufacturer Name	dmn	Name of manufacturer of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field containing the manufacturer name in the indicated language.	N	Manufacturer name	Name of the manufacturer as a ZigBee character string Defined in Basic Cluster	Y
Model Number	dmno	Model number as designated by manufacturer.	N	Model Identifier	Model number (or other identifier) assigned by the manufacturer as a ZigBee character string Defined in Basic Cluster	Y

1460 Table 38 shows how OCF Device Configuration properties, as specified in Table 15 in ISO/IEC
1461 30118-1:2018 shall be derived.

Table 38 – "oic.wk.con" Resource Type mapping

To OCF Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
(Device) Name	n	Human friendly name For example, "Bob's Thermostat"	Y	User description if it exists, else Model Name if it exists, else translate Application Device Identifier (=Device ID)to Human friendly name by using Application Device Identifier value/description table	User description : Information that allows the user to identify the device using a user-friendly character string, such as "Bedroom TV" Defined in User Descriptor Model Name : character string representing the name of the manufacturer's model of the device Defined in Complex Descriptor Application Device Identifier: device description supported on this endpoint Cluster Defined in Simple Descriptor	User description: N Model Name: N Application Device Identifier: Y
Location	loc	Provides location information where available.	N	(none)	(none)	
Location Name	locn	Human friendly name for location For example, "Living Room".	N	(none)	(none)	
Currency	c	Indicates the currency that is used for any monetary transactions	N	(none)	(none)	
Region	r	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").	N	(none)	(none)	
Localized Names	ln	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a "language" field (containing an RFC 5646 language tag) and a "value" field	N	User description if it exists, else Model Name if it exists, else translate Application Device Identifier (=Device ID)to	User description : Information that allows the user to identify the device using a user-friendly character string, such as "Bedroom TV" Defined in User Descriptor	User description: N Model Name: N Application Device Identifier: Y

To Property title	OCF Property name	OCF Description	OCF Mandatory	From Zigbee 3.0 Field name	Zigbee 3.0 Description	Zigbee 3.0 Mandatory
		containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.		Human friendly name by using Application Device Identifier value/description table	Model Name : character string representing the name of the manufacturer's model of the device Defined in Complex Descriptor Application Device Identifier: device description supported on this endpoint Cluster Defined in Simple Descriptor	
Default Language	dl	The default language supported by the Device, specified as an RFC 5646 language tag. By default, clients can treat any string property as being in this language unless the property specifies otherwise.	N	ISO 639-1 language code (if it exists, else property is absent)	Language used for character strings.	N

1463

1464 9 Device type definitions

1465 The required Resource Types are listed in Table 39.

1466

Table 39 – Device type definitions

Device Name (informative)	Device Type ("rt") (Normative)	Required Resource name	Required Resource Type
Bridge	oic.d.bridge	Secure Mode	oic.r.securemode
Virtual Device	oic.d.virtual	Device	oic.wk.d

1467 10 Resource type definitions

1468 10.1 List of resource types

1469 Table 40 lists the Resource Types defined in this document.

1470

Table 40 – Alphabetical list of resource types

Friendly Name (informative)	Resource Type (rt)	Clause
AllJoyn Object	oic.r.alljoynobject	10.2
Secure Mode	oic.r.securemode	10.3
VOD List	oic.r.vodlist	10.4

1471

1472 10.2 AllJoynObject

1473 10.2.1 Introduction

1474 This Resource is a Collection of Resources that were all derived from the same AllJoyn object.
1475

1476 10.2.2 Example URI

1477 /example/AllJoynObject

1478 10.2.3 Resource type

1479 The Resource Type is defined as: "oic.r.alljoynobject, oic.wk.col".

1480 10.2.4 OpenAPI 2.0 definition

```
1481 {  
1482   "swagger": "2.0",  
1483   "info": {  
1484     "title": "AllJoynObject",  
1485     "version": "2019-03-19",  
1486     "license": {  
1487       "name": "OCF Data Model License",  
1488       "url":  
1489         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
1490         CENSE.md",  
1491       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."  
1492     },  
1493     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
1494   },  
1495   "schemes": ["http"],  
1496   "consumes": ["application/json"],  
1497   "produces": ["application/json"],  
1498   "paths": {  
1499     "/example/AllJoynObject?if=oic.if.ll": {  
1500       "get": {  
1501         "description": "This Resource is a Collection of Resources that were all derived from the  
1502         same AllJoyn object.\n",  
1503         "parameters": [ { "$ref": "#/parameters/interface-all" } ],  
1504         "responses": {  
1505           "200": {  
1506             "description": "",  
1507             "x-example": [ {  
1508               "href": "/myRes1URI",  
1509               "rt": ["x.example.widget.false"],  
1510               "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],  
1511               "eps": [ { "ep": "coaps://[2001:db8:a::b1d4]:11111" } ]  
1512             } ],  
1513             "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],  
1514             "eps": [ { "ep": "coaps://[2001:db8:a::b1d4]:11111" } ]  
1515           }  
1516           },  
1517           {  
1518             "href": "/myRes2URI",  
1519             "rt": ["x.example.widget.true"],  
1520             "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],  
1521             "eps": [ { "ep": "coaps://[2001:db8:a::b1d4]:11111" } ]  
1522           }  
1523           },  
1524           {  
1525             "href": "/myRes3URI",  
1526             "rt": ["x.example.widget.method1"],  
1527             "if": ["oic.if.rw", "oic.if.baseline"],  
1528             "eps": [ { "ep": "coaps://[2001:db8:a::b1d4]:11111" } ]  
1529           }  
1530           }  
1531         }  
1532       }  
1533     }  
1534   }
```



```

1535         "href": "/myRes4URI",
1536         "rt": ["x.example.widget.method2"],
1537         "if": ["oic.if.rw", "oic.if.baseline"],
1538         "eps": [
1539             { "ep": "coaps://[2001:db8:a::b1d4]:11111" }
1540         ]
1541     },
1542 ],
1543     "schema": {
1544         "$ref": "#/definitions/slinks"
1545     }
1546 },
1547 },
1548 },
1549 },
1550 "/example/AllJoynObject?if=oic.if.baseline": {
1551     "get": {
1552         "description": "This Resource is a Collection of Resources that were all derived from the
1553 same AllJoyn object.\n",
1554         "parameters": [
1555             { "$ref": "#/parameters/interface-all" }
1556         ],
1557         "responses": {
1558             "200": {
1559                 "description": "",
1560                 "x-example": {
1561                     "rt": ["oic.r.alljoynobject", "oic.wk.col"],
1562                     "links": [
1563                         {
1564                             "href": "/myRes1URI",
1565                             "rt": ["x.example.widget.false"],
1566                             "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
1567                             "eps": [
1568                                 { "ep": "coaps://[2001:db8:a::b1d4]:11111" }
1569                             ]
1570                         },
1571                         {
1572                             "href": "/myRes2URI",
1573                             "rt": ["x.example.widget.true"],
1574                             "if": ["oic.if.r", "oic.if.rw", "oic.if.baseline"],
1575                             "eps": [
1576                                 { "ep": "coaps://[2001:db8:a::b1d4]:11111" }
1577                             ]
1578                         },
1579                         {
1580                             "href": "/myRes3URI",
1581                             "rt": ["x.example.widget.method1"],
1582                             "if": ["oic.if.rw", "oic.if.baseline"],
1583                             "eps": [
1584                                 { "ep": "coaps://[2001:db8:a::b1d4]:11111" }
1585                             ]
1586                         },
1587                         {
1588                             "href": "/myRes4URI",
1589                             "rt": ["x.example.widget.method2"],
1590                             "if": ["oic.if.rw", "oic.if.baseline"],
1591                             "eps": [
1592                                 { "ep": "coaps://[2001:db8:a::b1d4]:11111" }
1593                             ]
1594                         }
1595                     ]
1596                 },
1597                 "schema": {
1598                     "$ref": "#/definitions/AllJoynObject"
1599                 }
1600             }
1601         }
1602     }
1603 },
1604 },

```

```

1605     "parameters":
1606         "interface-all":
1607             "in":
1608                 "name":
1609                 "type":
1610                 "enum":
1611             }
1612         },
1613         "definitions":
1614         "oic.oic-link":
1615             "type":
1616             "properties":
1617                 "anchor":
1618                 "$ref":
1619 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1620 schema.json#/definitions/anchor"
1621             },
1622             "di":
1623                 "$ref":
1624 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1625 schema.json#/definitions/di"
1626             },
1627             "eps":
1628                 "$ref":
1629 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1630 schema.json#/definitions/eps"
1631             },
1632             "href":
1633                 "$ref":
1634 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1635 schema.json#/definitions/href"
1636             },
1637             "ins":
1638                 "$ref":
1639 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1640 schema.json#/definitions/ins"
1641             },
1642             "p":
1643                 "$ref":
1644 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1645 schema.json#/definitions/p"
1646             },
1647             "rel":
1648                 "$ref":
1649 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1650 schema.json#/definitions/rel_array"
1651             },
1652             "title":
1653                 "$ref":
1654 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1655 schema.json#/definitions/title"
1656             },
1657             "type":
1658                 "$ref":
1659 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
1660 schema.json#/definitions/type"
1661             },
1662             "if":
1663                 "description": "The OCF Interfaces supported by the target Resource",
1664                 "items":
1665                     "enum":
1666                         "oic.if.baseline",
1667                         "oic.if.ll",
1668                         "oic.if.r",
1669                         "oic.if.rw"
1670                     ],
1671                 "type":
1672                 "maxLength":
1673             },
1674             "minItems":

```

```

1675         "uniqueItems": true,
1676         "type": "array"
1677     },
1678     "rt": {
1679         "description": "Resource Type of the target Resource",
1680         "items": {
1681             "maxLength": 64,
1682             "type": "string"
1683         },
1684         "minItems": 1,
1685         "uniqueItems": true,
1686         "type": "array"
1687     }
1688 },
1689 "required": [
1690     "href",
1691     "rt",
1692     "if"
1693 ]
1694 },
1695 "slinks" : {
1696     "type": "array",
1697     "items": {
1698         "$ref": "#/definitions/oic.oic-link"
1699     }
1700 },
1701 "AllJoynObject" : {
1702     "type": "object",
1703     "properties": {
1704         "id": {
1705             "$ref":
1706 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
1707 schema.json#/definitions/id"
1708         },
1709         "if": {
1710             "description": "The interface set supported by this resource",
1711             "items": {
1712                 "enum": ["oic.if.baseline", "oic.if.ll"],
1713                 "type": "string"
1714             },
1715             "minItems": 1,
1716             "readOnly": true,
1717             "type": "array"
1718         },
1719         "n": {
1720             "$ref":
1721 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
1722 schema.json#/definitions/n"
1723         },
1724         "rt": {
1725             "items": {
1726                 "enum": ["oic.r.alljoynobject", "oic.wk.col"],
1727                 "type": "string"
1728             },
1729             "maxItems": 2,
1730             "minItems": 2,
1731             "uniqueItems": true,
1732             "readOnly": true,
1733             "type": "array"
1734         },
1735         "links" : {
1736             "type": "array",
1737             "description": "A set of OCF Links.",
1738             "items": {
1739                 "$ref": "#/definitions/oic.oic-link"
1740             }
1741         }
1742     }
1743 }
1744 }

```

1745 }
1746

1747 10.2.5 Property definition

1748 Table 41 defines the Properties that are part of the "oic.r.alljoynobject, oic.wk.col" Resource Type.

1749 **Table 41 – The Property definitions of the Resource with type "rt" = "oic.r.alljoynobject,**
1750 **oic.wk.col".**

Property name	Value type	Mandatory	Access mode	Description
id	multiple types: see schema		Read Write	
links	array: see schema		Read Write	A set of OCF Links.
n	multiple types: see schema		Read Write	
rt	array: see schema		Read Only	
if	array: see schema		Read Only	The interface set supported by this resource
rel	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the target Resource
p	multiple types: see schema	No	Read Write	
anchor	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Write	Resource Type of the target Resource
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
ins	multiple types: see schema	No	Read Write	
title	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	

1751 10.2.6 CRUDN behaviour

1752 Table 42 defines the CRUDN operations that are supported on the "oic.r.alljoynobject, oic.wk.col"
1753 Resource Type.

1754 **Table 42 – The CRUDN operations of the Resource with type "rt" = "oic.r.alljoynobject,**
1755 **oic.wk.col".**

Create	Read	Update	Delete	Notify
	get			observe

1756 10.3 SecureMode

1757 10.3.1 Introduction

1758 This Resource describes a secure mode on/off feature (on/off).

1759 A secureMode value of 'true' means that the feature is on, and any Bridged Server that cannot be
1760 communicated with securely shall not have a corresponding Virtual OCF Server, and any Bridged
1761 Client that cannot be communicated with securely shall not have a corresponding Virtual OCF
1762 Client.

1763 A secureMode value of 'false' means that the feature is off, any Bridged Server can have a
1764 corresponding Virtual OCF Server, and any Bridged Client can have a corresponding Virtual OCF
1765 Client.
1766

1767 10.3.2 Example URI

1768 /example/SecureModeResURI

1769 10.3.3 Resource type

1770 The Resource Type is defined as: "oic.r.securemode".

1771 10.3.4 OpenAPI 2.0 definition

```
1772 {
1773   "swagger": "2.0",
1774   "info": {
1775     "title": "SecureMode",
1776     "version": "2019-03-19",
1777     "license": {
1778       "name": "OCF Data Model License",
1779       "url": "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
1780 CENSE.md",
1781       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved.",
1782     },
1783     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
1784   },
1785   "schemes": ["http"],
1786   "consumes": ["application/json"],
1787   "produces": ["application/json"],
1788   "paths": {
1789     "/example/SecureModeResURI": {
1790       "get": {
1791         "description": "This Resource describes a secure mode on/off feature (on/off).\nA secureMode
1792 value of 'true' means that the feature is on, and any Bridged Server that cannot be communicated with
1793 securely shall not have a corresponding Virtual OCF Server, and any Bridged Client that cannot be
1794 communicated with securely shall not have a corresponding Virtual OCF Client.\nA secureMode value of
1795 'false' means that the feature is off, any Bridged Server can have a corresponding Virtual OCF Server,
1796 and any Bridged Client can have a corresponding Virtual OCF Client.\n",
1797         "parameters": [
1798           { "$ref": "#/parameters/interface" }
1799         ],
1800         "responses": {
1801           "200": {
1802             "description": "",
1803             "x-example": {
1804               "rt": "oic.r.securemode",
1805               "secureMode": false
1806             },
1807             "schema": {
1808               "$ref": "#/definitions/SecureMode"
1809             }
1810           }
1811         }
1812       }
1813     },
1814     "post": {
1815       "description": "Updates the value of secureMode.\n",
1816       "parameters": [
```

```

1817         {"$ref": "#/parameters/interface"},
1818         {
1819             "name": "body",
1820             "in": "body",
1821             "required": true,
1822             "schema": {
1823                 "$ref": "#/definitions/SecureMode-Update"
1824             },
1825             "x-example": {
1826                 "secureMode": true
1827             }
1828         }
1829     ],
1830     "responses": {
1831         "200": {
1832             "description": "",
1833             "x-example": {
1834                 "secureMode": true
1835             },
1836             "schema": {
1837                 "$ref": "#/definitions/SecureMode"
1838             }
1839         }
1840     }
1841 },
1842 },
1843 },
1844 "parameters": {
1845     "interface": {
1846         "in": "query",
1847         "name": "if",
1848         "type": "string",
1849         "enum": ["oic.if.rw", "oic.if.baseline"]
1850     }
1851 },
1852 "definitions": {
1853     "SecureMode": {
1854         "properties": {
1855             "id": {
1856                 "$ref":
1857 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
1858 schema.json#/definitions/id"
1859             },
1860             "if": {
1861                 "description": "The interface set supported by this resource",
1862                 "items": {
1863                     "enum": ["oic.if.baseline", "oic.if.rw"],
1864                     "type": "string",
1865                     "maxLength": 64
1866                 },
1867                 "minItems": 1,
1868                 "readOnly": true,
1869                 "uniqueItems": true,
1870                 "type": "array"
1871             },
1872             "n": {
1873                 "$ref":
1874 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
1875 schema.json#/definitions/n"
1876             },
1877             "rt": {
1878                 "description": "Resource Type",
1879                 "items": {
1880                     "enum": ["oic.r.securemode"],
1881                     "type": "string",
1882                     "maxLength": 64
1883                 },
1884                 "minItems": 1,
1885                 "uniqueItems": true,
1886                 "readOnly": true,

```

```

1887         "type": "array"
1888     },
1889     "secureMode": {
1890         "description": "Status of the Secure Mode",
1891         "type": "boolean"
1892     },
1893 },
1894 "required": ["secureMode"],
1895 "type": "object"
1896 },
1897 "SecureMode-Update": {
1898     "properties": {
1899         "secureMode": {
1900             "description": "Status of the Secure Mode",
1901             "type": "boolean"
1902         }
1903     }
1904 }
1905 }
1906 }
1907

```

10.3.5 Property definition

Table 43 defines the Properties that are part of the "oic.r.securemode" Resource Type.

Table 43 – The Property definitions of the Resource with type "rt" = "oic.r.securemode".

Property name	Value type	Mandatory	Access mode	Description
secureMode	boolean		Read Write	Status of the Secure Mode
secureMode	boolean	Yes	Read Write	Status of the Secure Mode
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this resource
rt	array: see schema	No	Read Only	Resource Type
id	multiple types: see schema	No	Read Write	

10.3.6 CRUDN behaviour

Table 44 defines the CRUDN operations that are supported on the "oic.r.securemode" Resource Type.

Table 44 – The CRUDN operations of the Resource with type "rt" = "oic.r.securemode".

Create	Read	Update	Delete	Notify
	get	post		observe

1915

10.4 VOD List

10.4.1 Introduction

This Resource describes the VODs that have been onboarded on the Bridge Platform.

1919

10.4.2 Example URI

/VODListResURI

1921

1922 10.4.3 Resource type

1923 The Resource Type is defined as: "oic.r.vodlist".

1924 10.4.4 OpenAPI 2.0 definition

```
1925 {
1926   "swagger": "2.0",
1927   "info": {
1928     "title": "VOD List",
1929     "version": "2019-05-16",
1930     "license": {
1931       "name": "OCF Data Model License",
1932       "url": "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
1933 CENSE.md",
1934       "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
1935     },
1936     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
1937   },
1938   "schemes": ["http"],
1939   "consumes": ["application/json"],
1940   "produces": ["application/json"],
1941   "paths": {
1942     "/VODListResURI": {
1943       "get": {
1944         "description": "This Resource describes the VODs that have been onboarded on the Bridge
1945 Platform.\n",
1946         "parameters": [
1947           {
1948             "$ref": "#/parameters/interface"
1949           }
1950         ],
1951         "responses": {
1952           "200": {
1953             "description": "Example response payload",
1954             "x-example": {
1955               "rt": ["oic.r.vodlist"],
1956               "vods": [
1957                 {
1958                   "n": "Smoke sensor",
1959                   "di": "54919CA5-4101-4AE4-595B-353C51AA1234",
1960                   "econame": "Z-Wave"
1961                 },
1962                 {
1963                   "n": "Thermostat",
1964                   "di": "54919CA5-4101-4AE4-595B-353C51AA5678",
1965                   "econame": "Zigbee"
1966                 }
1967               ]
1968             }
1969           },
1970           "schema": {
1971             "$ref": "#/definitions/vodlist"
1972           }
1973         }
1974       }
1975     },
1976     "parameters": {
1977       "interface": {
1978         "in": "query",
1979         "name": "if",
1980         "type": "string",
1981         "enum": ["oic.if.r", "oic.if.baseline"]
1982       }
1983     },
1984     "definitions": {
1985       "vodentry": {
1986         "description": "Information for a VOD created by the Bridge",
1987         "type": "object",
1988         "properties": {
1989           "n": {
```



```

1989         "$ref":
1990         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
1991         schema.json#/definitions/n"
1992     },
1993     "di" :
1994     {
1995         "$ref": "https://openconnectivityfoundation.github.io/core/schemas/oic.types-
1996         schema.json#/definitions/uuid"
1997     },
1998     "econame": {
1999         "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD",
2000         "type": "string",
2001         "enum": [ "BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave" ],
2002         "readOnly": true
2003     },
2004     "required": [ "n", "di", "econame" ]
2005 },
2006 "vodlist": {
2007     "type": "object",
2008     "properties": {
2009         "n": {
2010             "$ref":
2011             "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
2012             schema.json#/definitions/n"
2013         },
2014         "rt" :
2015         {
2016             "description": "Resource Type",
2017             "items": {
2018                 "maxLength": 64,
2019                 "type": "string",
2020                 "enum": [ "oic.r.vodlist" ]
2021             },
2022             "minItems": 1,
2023             "uniqueItems": true,
2024             "readOnly": true,
2025             "type": "array"
2026         },
2027         "id": {
2028             "$ref":
2029             "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
2030             schema.json#/definitions/id"
2031         },
2032         "if" :
2033         {
2034             "description": "The OCF Interface set supported by this Resource",
2035             "items": {
2036                 "enum": [
2037                     "oic.if.baseline",
2038                     "oic.if.r"
2039                 ],
2040                 "type": "string"
2041             },
2042             "minItems": 2,
2043             "uniqueItems": true,
2044             "readOnly": true,
2045             "type": "array"
2046         },
2047         "vods": {
2048             "description": "Array of information per VOD created by the Bridge",
2049             "type": "array",
2050             "minItems": 0,
2051             "uniqueItems": true,
2052             "readOnly": true,
2053             "items": {
2054                 "$ref": "#/definitions/vodentry"
2055             }
2056         }
2057     },
2058     "required": [ "vods" ]
}

```

2059 }
2060

2061 10.4.5 Property definition

2062 Table 45 defines the Properties that are part of the "oic.r.vodlist" Resource Type.

2063 **Table 45 – The Property definitions of the Resource with type "rt" = "oic.r.vodlist".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The OCF Interface set supported by this Resource
vods	array: see schema	Yes	Read Only	Array of information per VOD created by the Bridge
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type
econame	string	Yes	Read Only	Ecosystem Name of the Bridged Device which is exposed by this VOD
n	multiple types: see schema	Yes	Read Write	
di	multiple types: see schema	Yes	Read Write	

2064 10.4.6 CRUDN behaviour

2065 Table 46 defines the CRUDN operations that are supported on the "oic.r.vodlist" Resource Type.

2066 **Table 46 – The CRUDN operations of the Resource with type "rt" = "oic.r.vodlist".**

Create	Read	Update	Delete	Notify
	get			observe

2067
2068