

OCF Bridging Specification

VERSION 1.3.0 | November 2017



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

Copyright Open Connectivity Foundation, Inc. © 2017.
All Rights Reserved.

Legal Disclaimer

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

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

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

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

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

26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

CONTENTS

1	Scope	6
2	Normative references	6
3	Terms, definitions, symbols and abbreviations	7
3.1	Terms and definitions	7
3.2	Symbols and abbreviations	9
3.3	Conventions	9
4	Document conventions and organization	9
4.1	Notation.....	10
4.2	Data types	10
4.3	Document structure	10
5	Operational Scenarios.....	10
5.1	“Deep translation” vs. “on-the-fly”	11
5.2	Use of introspection.....	11
5.3	Stability and loss of data	11
6	OCF Bridge Device	12
6.1	Resource Discovery.....	13
6.2	General Requirements.....	22
6.3	Security.....	22
6.3.1	Blocking communication of Bridged Devices with the OCF ecosystem	23
7	AllJoyn Translation.....	23
7.1	Requirements Specific to an AllJoyn Translator	23
7.1.1	Exposing AllJoyn producer devices to OCF Clients	24
7.1.2	Exposing OCF resources to AllJoyn consumer applications	31
7.2	On-the-Fly Translation from D-Bus and OCF payloads.....	37
7.2.1	Translation without aid of introspection	37
7.2.2	Translation with aid of introspection	43
8	Device Type Definitions.....	48
9	Resource Type definitions	48
9.1	List of resource types	48
9.2	Secure Mode	48
9.2.1	Introduction	48
9.2.2	Example URI Path.....	49
9.2.3	Resource Type	49
9.2.4	RAML Definition	49
9.2.5	Swagger2.0 Definition	51
9.2.6	Property Definition	53
9.2.7	CRUDN behaviour.....	53
9.3	AllJoyn Object	53
9.3.1	Introduction	53

68	9.3.2	Example URI Path.....	53
69	9.3.3	Resource Type.....	53
70	9.3.4	RAML Definition.....	54
71	9.3.5	Swagger2.0 Definition.....	56
72	9.3.6	CRUDN behaviour.....	58
73			

Figures

74	
75	Figure 1. OCF Bridge Device Components 7
76	Figure 2: Schematic overview of an OCF Bridge Device bridging non-OCF devices 12
77	

Tables

78		
79	Table 1: oic.wk.d resource type definition	27
80	Table 2: oic.wk.con resource type definition	28
81	Table 3: oic.wk.p Resource Type definition.....	30
82	Table 4: oic.wk.con.p Resource Type definition	31
83	Table 5: AllJoyn About Data fields	33
84	Table 6: AllJoyn Configuration Data fields	36
85	Table 7 Alphabetical list of resource types.....	48
86		
87		

88 1 Scope

89 This document specifies a framework for translation between OCF devices and other ecosystems,
90 and specifies the behaviour of a translator that exposes AllJoyn producer applications to OCF
91 clients, and exposes OCF servers to AllJoyn consumer applications. Translation of specific AllJoyn
92 interfaces to or from specific OCF resource types is left to other specifications. Translation of
93 protocols other than AllJoyn is left to a future version of this specification. This document provides
94 generic requirements that apply unless overridden by a more specific document.

95 2 Normative references

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

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

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

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

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

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

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

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

113 IETF RFC 7049, *Concise Binary Object Representation (CBOR)*, October 2013
114 <https://www.rfc-editor.org/info/rfc7049>

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

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

119 JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January
120 2013
121 <http://json-schema.org/latest/json-schema-validation.html>

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

125 OCF Core Specification, *Open Connectivity Foundation Core Specification*, Version 1.3
126 Available at: https://openconnectivity.org/specs/OCF_Core_Specification_v1.3.0.pdf
127 Latest version available at: https://openconnectivity.org/specs/OCF_Core_Specification.pdf

128 OCF Security Specification, *Open Connectivity Foundation Security Specification*, Version 1.3
129 https://openconnectivity.org/specs/OCF_Security_Specification_v1.3.0.pdf
130 Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

131 OCF Resource to AllJoyn Interface Mapping Specification, *Open Connectivity Foundation*
132 *Resource to AllJoyn Interface Mapping Specification*, Version 1.3
133 Available at:
134 https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping_v1.3.0.pdf
135 Latest version available at:
136 https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping.pdf

137 OIC Core Specification, *Open Interconnect Consortium Core Specification*, Version 1.1
138 https://openconnectivity.org/specs/OIC_Core_Specification_v1.1.2.pdf

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

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

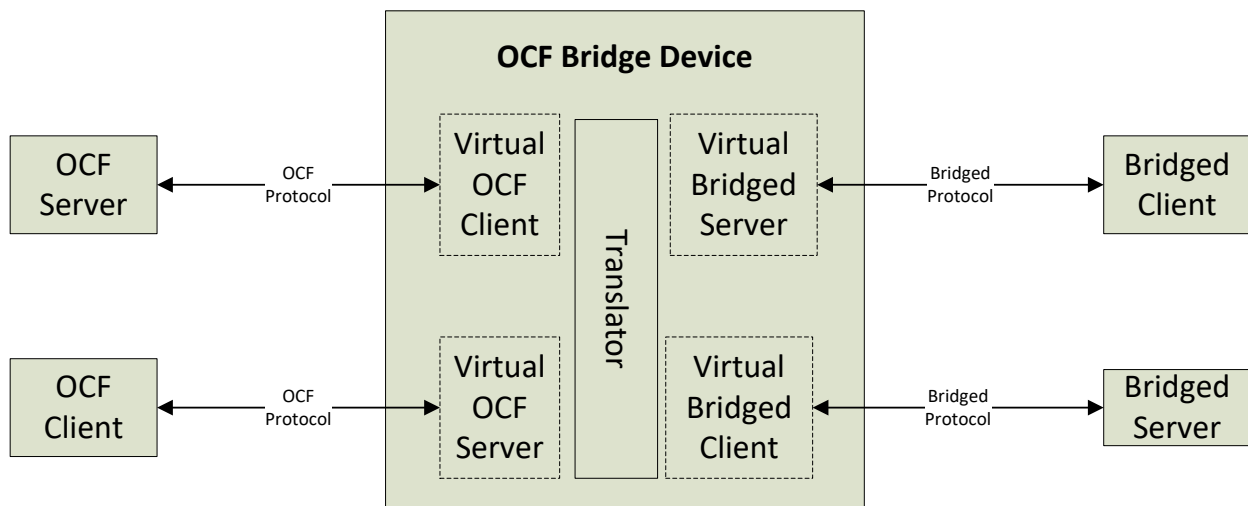
143 3 Terms, definitions, symbols and abbreviations

144 3.1 Terms and definitions

145 3.1.1

146 OCF Bridge Device

147 An OCF Device that can represent devices that exist on the network but communicate using a
148 Bridged Protocol rather than OCF protocols.



149
150
151

Figure 1. OCF Bridge Device Components

152 3.1.2

153 Bridged Protocol

154 another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

155 3.1.3

156 Translator

157 an OCF Bridge Device component that is responsible for translating to or from a specific Bridged
158 Protocol. More than one translator can exist on the same OCF Bridge Device, for different Bridged
159 Protocols.

160 **3.1.4**
161 **OCF Client**
162 a logical entity that accesses an OCF Resource on an OCF Server, which might be a Virtual OCF
163 Server exposed by the OCF Bridge Device.

164 **3.1.5**
165 **Bridged Client**
166 a logical entity that accesses data via a Bridged Protocol. For example, an AllJoyn Consumer
167 application is a Bridged Client.

168 **3.1.6**
169 **Virtual OCF Client**
170 a logical representation of a Bridged Client, which an OCF Bridge Device exposes to OCF Servers.

171 **3.1.7**
172 **Virtual Bridged Client**
173 a logical representation of an OCF Client, which an OCF Bridge Device exposes to Bridged Servers.

174 **3.1.8**
175 **OCF Device**
176 a logical entity that assumes one or more OCF roles (OCF Client, OCF Server). More than one
177 OCF Device can exist on the same physical platform.

178 **3.1.9**
179 **Virtual OCF Server**
180 a logical representation of a Bridged Server, which an OCF Bridge Device exposes to OCF Clients.

181 **3.1.10**
182 **Bridged Server**
183 a logical entity that provides data via a Bridged Protocol. For example, an AllJoyn Producer is a
184 Bridged Server. More than one Bridged Server can exist on the same physical platform.

185 **3.1.11**
186 **Virtual Bridged Server**
187 a logical representation of an OCF Server, which an OCF Bridge Device exposes to Bridged Clients.

188 **3.1.12**
189 **OCF Resource**
190 represents an artifact modelled and exposed by the OCF Framework

191 **3.1.13**
192 **Virtual OCF Resource**
193 a logical representation of a Bridged Resource, which an OCF Bridge Device exposes to OCF
194 Clients.

195 **3.1.14**
196 **Bridged Resource**
197 represents an artifact modelled and exposed by a Bridged Protocol. For example, an AllJoyn
198 object is a Bridged Resource.

199 **3.1.15**
200 **OCF Resource Property**
201 a significant aspect or notion including metadata that is exposed through the OCF Resource

202 **3.1.16**
203 **OCF Resource Type**
204 an OCF Resource Property that represents the data type definition for the OCF Resource

205 **3.1.17**
206 **Bridged Resource Type**
207 a schema used with a Bridged Protocol. For example, AllJoyn Interfaces are Bridged Resource
208 Types.

209 **3.1.18**
210 **OCF Server**
211 a logical entity with the role of providing resource state information and allowing remote control of
212 its resources.

213 **3.1.19**
214 **Onboarding Tool**
215 defined by the OCF Security Specification as: A logical entity within a specific IoT network that
216 establishes ownership for a specific device and helps bring the device into operational state within
217 that network.

218 **3.1.20**
219 **Bridged Device**
220 a Bridged Client or Bridged Server.

221 **3.1.21**
222 **Virtual OCF Device**
223 a Virtual OCF Client or Virtual OCF Server.

224 **3.2 Symbols and abbreviations**

225 **3.2.1**
226 **CRUDN**
227 Create Read Update Delete Notify
228 indicating which operations are possible on the resource

229 **3.2.2**
230 **CSV**
231 Comma Separated Value List
232 construction to have more fields in 1 string separated by commas. If a value contains a comma,
233 then the comma can be escaped by adding “\” in front of the comma.

234 **3.2.3**
235 **OCF**
236 Open Connectivity Foundation
237 organization that created these specifications

238 **3.2.4**
239 **RAML**
240 RESTful API Modeling Language
241 Simple and succinct way of describing practically RESTful APIs (see the RAML Specification)

242 **3.3 Conventions**
243 In this specification several terms, conditions, mechanisms, sequences, parameters, events,
244 states, or similar terms are printed with the first letter of each word in uppercase and the rest
245 lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal
246 technical English meaning.

247 **4 Document conventions and organization**
248 For the purposes of this document, the terms and definitions given in the OCF 1.0 Core
249 Specification apply.

250 **4.1 Notation**

251 In this document, features are described as required, recommended, allowed or DEPRECATED as
252 follows:

253 Required (or shall or mandatory).

- 254 – These basic features shall be implemented to comply with this specification. The phrases “shall
255 not”, and “PROHIBITED” indicate behaviour that is prohibited, i.e. that if performed means the
256 implementation is not in compliance.

257 Recommended (or should).

- 258 – These features add functionality supported by this specification and should be implemented.
259 Recommended features take advantage of the capabilities of this specification, usually without
260 imposing major increase of complexity. Notice that for compliance testing, if a recommended
261 feature is implemented, it shall meet the specified requirements to be in compliance with these
262 guidelines. Some recommended features could become requirements in the future. The phrase
263 “should not” indicates behaviour that is permitted but not recommended.

264 Allowed (or allowed).

- 265 – These features are neither required nor recommended, but if the feature is implemented, it
266 shall meet the specified requirements to be in compliance with these guidelines.

267 Conditionally allowed (CA)

- 268 – The definition or behaviour depends on a condition. If the specified condition is met, then the
269 definition or behaviour is allowed, otherwise it is not allowed.

270 Conditionally required (CR)

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

274 DEPRECATED

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

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

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

283 **4.2 Data types**

284 Data types are defined in the OCF 1.0 Core Specification.

285 **4.3 Document structure**

286 Section 5 discusses operational scenarios. Section 6 covers generic requirements for any OCF
287 Bridge, and section 7 covers the specific requirements for a Bridge that translates to/from AllJoyn.
288 These are covered separately to ease the task of defining translation to other protocols in the
289 future.

290 **5 Operational Scenarios**

291 The overall goals are to:

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

294 **5.1 “Deep translation” vs. “on-the-fly”**

295 When translating a service between a Bridged Protocol (e.g., AllJoyn) and OCF protocols, there
 296 are two possible types of translation. Translators are expected to dedicate most of their logic to
 297 “deep translation” types of communication, in which data models used with the Bridged Protocol
 298 are mapped to the equivalent OCF Resource Types and vice-versa, in such a way that a compliant
 299 OCF Client or Bridged Client would be able to interact with the service without realising that a
 300 translation was made.

301 “Deep translation” is out of the scope of this document, as the procedure far exceeds mapping of
 302 types. For example, clients on one side of a translator may decide to represent an intensity as an
 303 8-bit value between 0 and 255, whereas the devices on the other may have chosen to represent
 304 that as a floating-point number between 0.0 and 1.0. It’s also possible that the procedure may
 305 require storing state in the translator. Either way, the programming of such translation will require
 306 dedicated effort and study of the mechanisms on both sides.

307 The other type of translation, the “on-the-fly” or “one-to-one” translation, requires no prior
 308 knowledge of the device-specific schema in question on the part of the translator. The burden is,
 309 instead, on one of the other participants in the communication, usually the client application. That
 310 stems from the fact that “on-the-fly” translation always produces Bridged Resource Types and OCF
 311 Resource Types as *vendor extensions*.

312 For AllJoyn, deep translation is specified in OCF ASA Mapping, and on-the-fly translation is
 313 covered in section 7.2 of this document.

314 **5.2 Use of introspection**

315 Whenever possible, the translation code should make use of metadata available that indicates
 316 what the sender and recipient of the message in question are expecting. For example, devices that
 317 are AllJoyn Certified are required to carry the introspection data for each object and interface they
 318 expose. The OIC 1.1 Core Specification makes no such requirement, but the OCF 1.0 Core
 319 Specification does. When the metadata is available, translators should convert the incoming
 320 payload to exactly the format expected by the recipient and should use information when
 321 translating replies to form a more useful message.

322 For example, for an AllJoyn translator, the expected interaction list is presented on the list below:

Message Type	Sender	Receiver	Metadata
Request	AllJoyn 16.10	OIC 1.1	Not available
Request	AllJoyn 16.10	OCF 1.0	Available
Request	OIC 1.1 or OCF 1.0	AllJoyn 16.10	Available
Response	AllJoyn 16.10	OIC 1.1 or OCF 1.0	Available
Response	OIC 1.1	AllJoyn 16.10	Not available
Response	OCF 1.0	AllJoyn 16.10	Available

323 **5.3 Stability and loss of data**

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

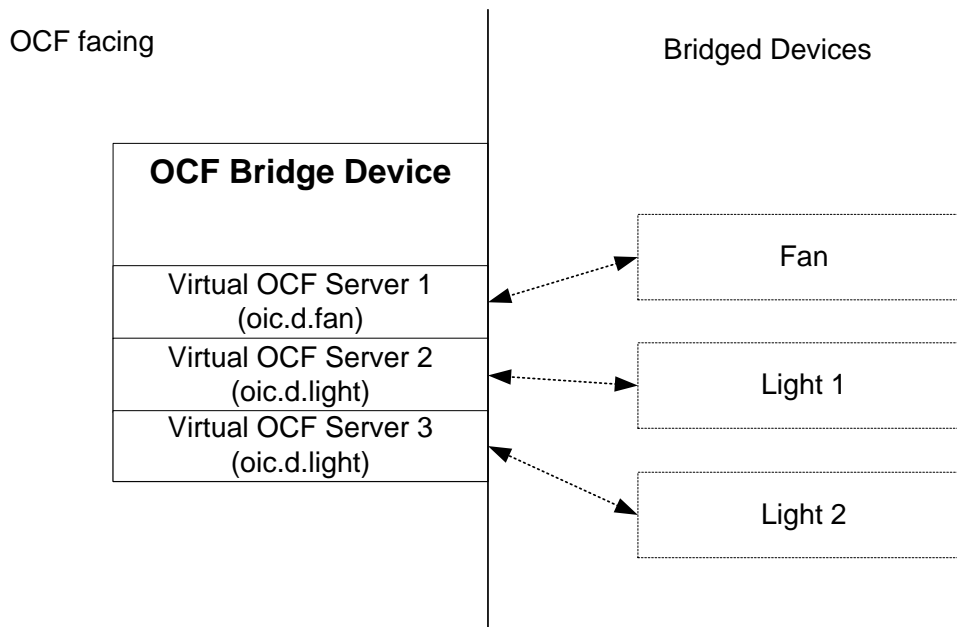
328 However, a third round of translation should produce the same identical message as was
329 previously produced, provided the same information is available. That is, in the above chain,
330 payloads 2 and 4 as well as 3 and 5 should be identical.

331 6 OCF Bridge Device

332 This section describes the functionality of an OCF Bridge Device; such a device is illustrated in
333 Figure 2.

334 An OCF Bridge Device is a device that represents one or more Bridged Devices as Virtual OCF
335 Devices on the network and/or represents one or more OCF Devices as Virtual Devices using
336 another protocol on the network. The Bridged Devices themselves are out of the scope of this
337 document. The only difference between a native OCF Device and a Virtual Bridged Device is how
338 the device is encapsulated in an OCF Bridge Device.

339 An OCF Bridge Device shall be indicated on the OCF network with a Device Type of "oic.d.bridge".
340 This provides to an OCF Client an explicit indication that the discovered Device is performing a
341 bridging function. This is useful for several reasons; 1) when establishing a home network the
342 Client can determine that the bridge is reachable and functional when no bridged devices are
343 present, 2) allows for specific actions to be performed on the bridge considering the known
344 functionality a bridge supports, 3) allows for explicit discovery of all devices that are serving a
345 bridging function which benefits trouble shooting and maintenance actions on behalf of a user.
346 When such a device is discovered the exposed Resources on the OCF Bridge Device describe
347 other devices. For example, as shown in Figure 2.



348

349 **Figure 2: Schematic overview of an OCF Bridge Device bridging non-OCF devices**

350 It is expected that the OCF Bridge Device creates a set of devices during the start-up of the OCF
351 Bridge Device. The exposed set of Virtual OCF Devices can change as Bridged Devices are added
352 or removed from the bridge. The adding and removing of Bridged Devices is implementation

353 dependent. When an OCF Bridge Device changes the set of exposed Virtual OCF Devices, it shall
354 notify any OCF Clients subscribed to its "/oic/res".

355 **6.1 Resource Discovery**

356 An OCF Bridge Device shall detect devices that arrive and leave the Bridged network or the OCF
357 network. Where there is no pre-existing mechanism to reliably detect the arrival and departure of
358 devices on a network, an OCF Bridge Device shall periodically poll the network to detect arrival
359 and departure of devices, for example using COAP multicast discovery (a multicast RETRIEVE of
360 "/oic/res") in the case of the OCF network. OCF Bridge Device implementations are encouraged to
361 use a poll interval of 30 seconds plus or minus a random delay of a few seconds.

362 An OCF Bridge Device shall respond to network discovery commands on behalf of the exposed
363 bridged devices. All bridged devices with all their Resources shall be listed in "/oic/res" of the
364 Bridge. The response to a RETRIEVE on "/oic/res" shall only include the devices that match the
365 RETRIEVE request.

366 The resource reference determined from each Link exposed by "/oic/res" on the Bridge shall be
367 unique. The Bridge shall meet the requirements defined in the OCF 1.0 Core Specification for
368 population of the Properties and Link parameters in "/oic/res".

369 For example, if an OCF Bridge Device exposes Virtual OCF Servers for the fan and lights shown
370 in Figure 2, the bridge might return the following information corresponding to the JSON below to
371 a legacy OIC 1.1 client doing a RETRIEVE on "/oic/res". (Note that what is returned is not in the
372 JSON format but in a suitable encoding as defined in the OCF 1.0 Core Specification.)

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

```
[
  {
    "di": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "links": [
      {
        "href": "coap://[2001:db8:a::b1d4]:55555/oic/res",
        "rel": "self",
        "rt": ["oic.wk.res"],
        "if": ["oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/d",
        "rt": ["oic.wk.d", "oic.d.bridge"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/p",
        "rt": ["oic.wk.p"],
        "if": ["oic.if.r", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/mySecureMode",
        "rt": ["oic.r.securemode"],
        "if": ["oic.if.rw", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 11111}
      },
      {
        "href": "/oic/sec/doxm",
        "rt": ["oic.r.doxm"],
        "if": ["oic.if.baseline"],
        "p": {"bm": 1, "sec": true, "port": 11111}
      }
    ]
  }
]
```

```

408     },
409     {
410         "href": "/oic/sec/pstat",
411         "rt": ["oic.r.pstat"],
412         "if": ["oic.if.baseline"],
413         "p": {"bm": 1, "sec": true, "port": 11111}
414     },
415     {
416         "href": "/oic/sec/cred",
417         "rt": ["oic.r.cred"],
418         "if": ["oic.if.baseline"],
419         "p": {"bm": 1, "sec": true, "port": 11111}
420     },
421     {
422         "href": "/oic/sec/acl2",
423         "rt": ["oic.r.acl2"],
424         "if": ["oic.if.baseline"],
425         "p": {"bm": 1, "sec": true, "port": 11111}
426     },
427     {
428         "href": "/myIntrospection",
429         "rt": ["oic.wk.introspection"],
430         "if": ["oic.if.r", "oic.if.baseline"],
431         "p": {"bm": 3, "sec": true, "port": 11111}
432     }
433 ]
434 },
435 {
436     "di": "88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
437     "links": [
438         {
439             "href": "coaps://[2001:db8:a::b1d4]:22222/oic/res",
440             "rt": ["oic.wk.res"],
441             "if": ["oic.if.ll", "oic.if.baseline"],
442             "p": {"bm": 3, "sec": true, "port": 22222}
443         },
444         {
445             "href": "coaps://[2001:db8:a::b1d4]:22222/oic/d",
446             "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
447             "if": ["oic.if.r", "oic.if.baseline"],
448             "p": {"bm": 3, "sec": true, "port": 22222}
449         },
450         {
451             "href": "coaps://[2001:db8:a::b1d4]:22222/oic/p",
452             "rt": ["oic.wk.p"],
453             "if": ["oic.if.r", "oic.if.baseline"],
454             "p": {"bm": 3, "sec": true, "port": 22222}
455         },
456         {
457             "href": "coaps://[2001:db8:a::b1d4]:22222/myFan",
458             "rt": ["oic.r.switch.binary"],
459             "if": ["oic.if.a", "oic.if.baseline"],
460             "p": {"bm": 3, "sec": true, "port": 22222}
461         },
462         {
463             "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/doxm",
464             "rt": ["oic.r.doxm"],
465             "if": ["oic.if.baseline"],
466             "p": {"bm": 1, "sec": true, "port": 22222}
467         },
468         {
469             "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/pstat",
470             "rt": ["oic.r.pstat"],

```

```

471     "if": ["oic.if.baseline"],
472     "p": {"bm": 1, "sec": true, "port": 22222}
473   },
474   {
475     "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/cred",
476     "rt": ["oic.r.cred"],
477     "if": ["oic.if.baseline"],
478     "p": {"bm": 1, "sec": true, "port": 22222}
479   },
480   {
481     "href": "coaps://[2001:db8:a::b1d4]:22222/oic/sec/acl2",
482     "rt": ["oic.r.acl2"],
483     "if": ["oic.if.baseline"],
484     "p": {"bm": 1, "sec": true, "port": 22222}
485   },
486   {
487     "href": "coaps://[2001:db8:a::b1d4]:22222/myFanIntrospection",
488     "rt": ["oic.wk.introspection"],
489     "if": ["oic.if.r", "oic.if.baseline"],
490     "p": {"bm": 3, "sec": true, "port": 22222}
491   }
492 ]
493 },
494 {
495   "di": "dc70373c-1e8d-4fb3-962e-017eaa863989",
496   "links": [
497     {
498       "href": "coaps://[2001:db8:a::b1d4]:33333/oic/res",
499       "rt": ["oic.wk.res"],
500       "if": ["oic.if.ll", "oic.if.baseline"],
501       "p": {"bm": 3, "sec": true, "port": 33333}
502     },
503     {
504       "href": "coaps://[2001:db8:a::b1d4]:33333/oic/d",
505       "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
506       "if": ["oic.if.r", "oic.if.baseline"],
507       "p": {"bm": 3, "sec": true, "port": 33333}
508     },
509     {
510       "href": "coaps://[2001:db8:a::b1d4]:33333/oic/p",
511       "rt": ["oic.wk.p"],
512       "if": ["oic.if.r", "oic.if.baseline"],
513       "p": {"bm": 3, "sec": true, "port": 33333}
514     },
515     {
516       "href": "coaps://[2001:db8:a::b1d4]:33333/myLight",
517       "rt": ["oic.r.switch.binary"],
518       "if": ["oic.if.a", "oic.if.baseline"],
519       "p": {"bm": 3, "sec": true, "port": 33333}
520     },
521     {
522       "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/doxm",
523       "rt": ["oic.r.doxm"],
524       "if": ["oic.if.baseline"],
525       "p": {"bm": 1, "sec": true, "port": 33333}
526     },
527     {
528       "href": "coaps://[2001:db8:a::b1d4]:33333/oic/sec/pstat",
529       "rt": ["oic.r.pstat"],
530       "if": ["oic.if.baseline"],
531       "p": {"bm": 1, "sec": true, "port": 33333}
532     }
533   ]

```



```

534     "href": "coaps://[2001:db8:a:b1d4]:33333/oic/sec/cred",
535     "rt": ["oic.r.cred"],
536     "if": ["oic.if.baseline"],
537     "p": {"bm": 1, "sec": true, "port": 33333}
538   },
539   {
540     "href": "coaps://[2001:db8:a:b1d4]:33333/oic/sec/acl2",
541     "rt": ["oic.r.acl2"],
542     "if": ["oic.if.baseline"],
543     "p": {"bm": 1, "sec": true, "port": 33333}
544   },
545   {
546     "href": "coaps://[2001:db8:a:b1d4]:33333/myLightIntrospection",
547     "rt": ["oic.wk.introspection"],
548     "if": ["oic.if.r", "oic.if.baseline"],
549     "p": {"bm": 3, "sec": true, "port": 33333}
550   }
551 ]
552 },
553 {
554   "di": "8202138e-aa22-452c-b512-9ebad02bef7c",
555   "links": [
556     {
557       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/res",
558       "rt": ["oic.wk.res"],
559       "if": ["oic.if.ll", "oic.if.baseline"],
560       "p": {"bm": 3, "sec": true, "port": 44444}
561     },
562     {
563       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/d",
564       "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
565       "if": ["oic.if.r", "oic.if.baseline"],
566       "p": {"bm": 3, "sec": true, "port": 44444}
567     },
568     {
569       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/p",
570       "rt": ["oic.wk.p"],
571       "if": ["oic.if.r", "oic.if.baseline"],
572       "p": {"bm": 3, "sec": true, "port": 44444}
573     },
574     {
575       "href": "coaps://[2001:db8:a:b1d4]:44444/myLight",
576       "rt": ["oic.r.switch.binary"],
577       "if": ["oic.if.a", "oic.if.baseline"],
578       "p": {"bm": 3, "sec": true, "port": 44444}
579     },
580     {
581       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/sec/doxm",
582       "rt": ["oic.r.doxm"],
583       "if": ["oic.if.baseline"],
584       "p": {"bm": 1, "sec": true, "port": 44444}
585     },
586     {
587       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/sec/pstat",
588       "rt": ["oic.r.pstat"],
589       "if": ["oic.if.baseline"],
590       "p": {"bm": 1, "sec": true, "port": 44444}
591     },
592     {
593       "href": "coaps://[2001:db8:a:b1d4]:44444/oic/sec/cred",
594       "rt": ["oic.r.cred"],
595       "if": ["oic.if.baseline"],
596       "p": {"bm": 1, "sec": true, "port": 44444}

```

```

597     },
598     {
599         "href": "coaps://[2001:db8:a::b1d4]:44444/oic/sec/acl2",
600         "rt": ["oic.r.acl2"],
601         "if": ["oic.if.baseline"],
602         "p": {"bm": 1, "sec": true, "port": 44444}
603     },
604     {
605         "href": "coaps://[2001:db8:a::b1d4]:44444/myLightIntrospection",
606         "rt": ["oic.wk.introspection"],
607         "if": ["oic.if.r", "oic.if.baseline"],
608         "p": {"bm": 3, "sec": true, "port": 44444}
609     }
610 ]
611 }
612 ]

```

613 The above example illustrates that each Virtual OCF Server has its own “di” and endpoint
614 exposed by the bridge, and that “/oic/p” and “/oic/d” are available for each Virtual OCF Server.

615
616 When an OCF Client requests a content format of “application/vnd.ocf+cbor”, the same bridge
617 will return information corresponding to the JSON below. (Note that what is returned is not in the
618 JSON format but in a suitable encoding as defined in the OCF 1.0 Core Specification.)

```

620 [
621 {
622     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
623     "href": "/oic/res",
624     "rel": "self",
625     "rt": ["oic.wk.res"],
626     "if": ["oic.if.ll", "oic.if.baseline"],
627     "p": {"bm": 3},
628     "eps": [{"ep": "coap://[2001:db8:a::b1d4]:55555"},
629             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
630 },
631 {
632     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
633     "href": "/oic/d",
634     "rt": ["oic.wk.d", "oic.d.bridge"],
635     "if": ["oic.if.r", "oic.if.baseline"],
636     "p": {"bm": 3},
637     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
638 },
639 {
640     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
641     "href": "/oic/p",
642     "rt": ["oic.wk.p"],
643     "if": ["oic.if.r", "oic.if.baseline"],
644     "p": {"bm": 3},
645     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
646 },
647 {
648     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
649     "href": "/mySecureMode",
650     "rt": ["oic.r.securemode"],
651     "if": ["oic.if.rw", "oic.if.baseline"],
652     "p": {"bm": 3},
653     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
654 },
655 {
656     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
657     "href": "/oic/sec/doxm",

```

```

658     "rt": ["oic.r.doxm"],
659     "if": ["oic.if.baseline"],
660     "p": {"bm": 1},
661     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
662   },
663   {
664     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
665     "href": "/oic/sec/pstat",
666     "rt": ["oic.r.pstat"],
667     "if": ["oic.if.baseline"],
668     "p": {"bm": 1},
669     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
670   },
671   {
672     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
673     "href": "/oic/sec/cred",
674     "rt": ["oic.r.cred"],
675     "if": ["oic.if.baseline"],
676     "p": {"bm": 1},
677     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
678   },
679   {
680     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
681     "href": "/oic/sec/acl2",
682     "rt": ["oic.r.acl2"],
683     "if": ["oic.if.baseline"],
684     "p": {"bm": 1},
685     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
686   },
687   {
688     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
689     "href": "/myIntrospection",
690     "rt": ["oic.wk.introspection"],
691     "if": ["oic.if.r", "oic.if.baseline"],
692     "p": {"bm": 3},
693     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
694   },
695
696
697   {
698     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
699     "href": "/oic/res",
700     "rt": ["oic.wk.res"],
701     "if": ["oic.if.ll", "oic.if.baseline"],
702     "p": {"bm": 3},
703     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
704   },
705   {
706     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
707     "href": "/oic/d",
708     "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
709     "if": ["oic.if.r", "oic.if.baseline"],
710     "p": {"bm": 3},
711     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
712   },
713   {
714     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
715     "href": "/oic/p",
716     "rt": ["oic.wk.p"],
717     "if": ["oic.if.r", "oic.if.baseline"],
718     "p": {"bm": 3},
719     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
720   },

```

```

721 {
722   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
723   "href": "/myFan",
724   "rt": ["oic.r.switch.binary"],
725   "if": ["oic.if.a", "oic.if.baseline"],
726   "p": {"bm": 3},
727   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
728 },
729 {
730   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
731   "href": "/oic/sec/doxm",
732   "rt": ["oic.r.doxm"],
733   "if": ["oic.if.baseline"],
734   "p": {"bm": 1},
735   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
736 },
737 {
738   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
739   "href": "/oic/sec/pstat",
740   "rt": ["oic.r.pstat"],
741   "if": ["oic.if.baseline"],
742   "p": {"bm": 1},
743   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
744 },
745 {
746   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
747   "href": "/oic/sec/cred",
748   "rt": ["oic.r.cred"],
749   "if": ["oic.if.baseline"],
750   "p": {"bm": 1},
751   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
752 },
753 {
754   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
755   "href": "/oic/sec/acl2",
756   "rt": ["oic.r.acl2"],
757   "if": ["oic.if.baseline"],
758   "p": {"bm": 1},
759   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
760 },
761 {
762   "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
763   "href": "/myFanIntrospection",
764   "rt": ["oic.wk.introspection"],
765   "if": ["oic.if.r", "oic.if.baseline"],
766   "p": {"bm": 3},
767   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
768 },
769 {
770   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
771   "href": "/oic/res",
772   "rt": ["oic.wk.res"],
773   "if": ["oic.if.ll", "oic.if.baseline"],
774   "p": {"bm": 3},
775   "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
776 },
777 {
778   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
779   "href": "/oic/d",
780   "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
781   "if": ["oic.if.r", "oic.if.baseline"],
782   "p": {"bm": 3},
783

```

```

784     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
785   },
786   {
787     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
788     "href": "/oic/p",
789     "rt": ["oic.wk.p"],
790     "if": ["oic.if.r", "oic.if.baseline"],
791     "p": {"bm": 3},
792     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
793   },
794   {
795     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
796     "href": "/myLight",
797     "rt": ["oic.r.switch.binary"],
798     "if": ["oic.if.a", "oic.if.baseline"],
799     "p": {"bm": 3},
800     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
801   },
802   {
803     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
804     "href": "/oic/sec/doxm",
805     "rt": ["oic.r.doxm"],
806     "if": ["oic.if.baseline"],
807     "p": {"bm": 1},
808     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
809   },
810   {
811     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
812     "href": "/oic/sec/pstat",
813     "rt": ["oic.r.pstat"],
814     "if": ["oic.if.baseline"],
815     "p": {"bm": 1},
816     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
817   },
818   {
819     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
820     "href": "/oic/sec/cred",
821     "rt": ["oic.r.cred"],
822     "if": ["oic.if.baseline"],
823     "p": {"bm": 1},
824     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
825   },
826   {
827     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
828     "href": "/oic/sec/acl2",
829     "rt": ["oic.r.acl2"],
830     "if": ["oic.if.baseline"],
831     "p": {"bm": 1},
832     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
833   },
834   {
835     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
836     "href": "/myLightIntrospection",
837     "rt": ["oic.wk.introspection"],
838     "if": ["oic.if.r", "oic.if.baseline"],
839     "p": {"bm": 3},
840     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
841   },
842   {
843     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
844     "href": "/oic/res",
845     "rt": ["oic.wk.res"],
846

```

```

847     "if": ["oic.if.ll", "oic.if.baseline"],
848     "p": {"bm": 3},
849     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
850   },
851   {
852     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
853     "href": "/oic/d",
854     "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
855     "if": ["oic.if.r", "oic.if.baseline"],
856     "p": {"bm": 3},
857     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
858   },
859   {
860     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
861     "href": "/oic/p",
862     "rt": ["oic.wk.p"],
863     "if": ["oic.if.r", "oic.if.baseline"],
864     "p": {"bm": 3},
865     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
866   },
867   {
868     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
869     "href": "/myLight",
870     "rt": ["oic.r.switch.binary"],
871     "if": ["oic.if.a", "oic.if.baseline"],
872     "p": {"bm": 3},
873     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
874   },
875   {
876     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
877     "href": "/oic/sec/doxm",
878     "rt": ["oic.r.doxm"],
879     "if": ["oic.if.baseline"],
880     "p": {"bm": 1},
881     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
882   },
883   {
884     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
885     "href": "/oic/sec/pstat",
886     "rt": ["oic.r.pstat"],
887     "if": ["oic.if.baseline"],
888     "p": {"bm": 1},
889     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
890   },
891   {
892     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
893     "href": "/oic/sec/cred",
894     "rt": ["oic.r.cred"],
895     "if": ["oic.if.baseline"],
896     "p": {"bm": 1},
897     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
898   },
899   {
900     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
901     "href": "/oic/sec/acl2",
902     "rt": ["oic.r.acl2"],
903     "if": ["oic.if.baseline"],
904     "p": {"bm": 1},
905     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
906   },
907   {
908     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
909     "href": "/myLightIntrospection",

```

```
910     "rt": ["oic.wk.introspection"],
911     "if": ["oic.if.r", "oic.if.baseline"],
912     "p": {"bm": 3},
913     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
914   }
915 ]
```

916 6.2 General Requirements

917
918 The translator shall check the protocol-independent UUID (“piid” in OCF) of each device and shall
919 not advertise back into a Bridged Protocol a device originally seen via that Bridged Protocol. The
920 translator shall stop translating any Bridged Protocol device exposed in OCF via another translator
921 if the translator sees the device via the Bridged Protocol. Similarly, the translator shall not
922 advertise an OCF Device back into OCF, and the translator shall stop translating any OCF device
923 exposed in the Bridged Protocol via another translator if the translator sees the device via OCF.
924 These require that the translator can determine when a device is already being translated. A
925 Virtual OCF Device shall be indicated on the OCF network with a Device Type of “oic.d.virtual”.
926 This allows translators to determine if a device is already being translated when multiple translators
927 are present. How a translator determines if a device is already being translated on a non-OCF
928 network is described in the protocol-specific sections below.

929
930 The translator shall detect duplicate virtual devices (with the same protocol-independent UUID)
931 present in a network and shall not create more than one corresponding virtual device as it
932 translates those duplicate devices into another network.

933
934 Each Bridged Server shall be exposed as a separate Virtual OCF Server, with its own endpoint,
935 and its own “/oic/d” and “/oic/p”. The Virtual OCF Server’s “/oic/res” resource would be the same
936 as for any ordinary OCF Server that uses a resource directory. That is, it does not respond to
937 multicast discovery requests (because the OCF Bridge Device responds on its behalf), but a
938 unicast query elicits a response listing its own resources with a “rel”=“hosts” relationship, and an
939 appropriate “anchor” to indicate that it is not the OCF Bridge Device itself. This allows platform-
940 specific, device-specific, and resource-specific fields to all be preserved across translation.

941
942 The introspection data provided by the translator shall include information about all the virtual
943 devices (and their resources) exposed by the translator at that point in time. This means that the
944 introspection data provided by the translator before and after a new virtual device is exposed would
945 be different.

946 6.3 Security

947 The OCF Bridge Device shall go through OCF ownership transfer as any other onboarder would.
948 Separately, it shall go through the Bridged Protocol’s ownership transfer mechanism (e.g., AllJoyn
949 claiming) normally as any other onboarder would.

950
951 The OCF Bridge Device shall be field updatable. (This requirement need not be tested but can
952 be certified via a vendor declaration.)

953
954 Unless an administrator opts in to allow it (see section 9.2), a translator shall not expose
955 connectivity to devices that it cannot get a secure connection to.

956 Each Virtual OCF Device shall be provisioned for security by an OCF Onboarding tool. Each Virtual
957 Bridged Device should be provisioned as appropriate in the Bridged ecosystem. In other words,
958 Virtual Devices are treated the same way as physical Devices. They are entities that have to be
959 provisioned in their network.

960 The Translator shall provide a “piid” value that can be used to correlate a non-OCF Device with its
961 corresponding Virtual OCF Device, as specified in Section 6.2. An Onboarding Tool might use this
962 correlation to improve the Onboarding user experience by eliminating or reducing the need for user

963 input, by automatically creating security settings for Virtual OCF Devices that are equivalent to the
964 security settings of their corresponding non-OCF Devices. See the OCF Security Specification for
965 detailed information about Onboarding.

966 Each Virtual Device shall implement the security requirements of the ecosystem that it is connected
967 to. For example, each Virtual OCF Device shall implement the behaviour required by the OCF 1.0
968 Core Specification and the OCF Security Specification. Each Virtual OCF Device shall perform
969 authentication, access control, and encryption according to the security settings it received from
970 the Onboarding Tool.

971 Depending on the architecture of the Translator, authentication and access control might take
972 place just within each ecosystem, but not within the Translator. For example, when an OCF Client
973 sends a request to a Virtual OCF Server:

- 974 - Authentication and access control might be performed by the Virtual OCF Server when
975 receiving the request from the OCF Client.
- 976 - The Translator might not perform authentication or access control when the request travels
977 through the Translator to the corresponding Virtual Bridged Client.
- 978 - Authentication and access control might be performed by the target Bridged Server when
979 it receives the request from the Virtual Bridged Client, according to the security model of
980 the Bridged ecosystem.

981 A Translator may receive unencrypted data coming from a Bridged Client through a Virtual Bridged
982 Device. The translated message shall be encrypted by the corresponding Virtual OCF Client,
983 before sending it to the target OCF Device, if this OCF Device requires encryption.

984 A Translator may receive unencrypted data coming from an OCF Client through a Virtual OCF
985 Server. After translation, this data shall be encrypted by the corresponding Virtual Bridged Client,
986 before sending it to the target Bridged Server, if this Bridged Server requires encryption.

987 A Translator shall protect the data while that data travels between a Virtual Client and a Virtual
988 Server, through the Translator. For example, if the Translator sends data over a network, the
989 Translator shall perform appropriate authentication and access control, and shall encrypt the data,
990 between all peers involved in this communication.

991 **6.3.1 Blocking communication of Bridged Devices with the OCF ecosystem**

992 An OCF Onboarding Tool shall be able to block the communication of all OCF Devices with all
993 Bridged Devices that don't communicate securely with the Bridge, by using the Bridge Device's
994 "oic.r.securemode" Resource.

995 In addition, an OCF Onboarding Tool can block the communication of a particular Virtual OCF
996 Client with all OCF Servers, or block the communication of all OCF Clients with a particular Virtual
997 OCF Server, in the same way as it would for any other OCF Device. See section 8.5 of the OCF
998 Security Specification for information about the soft reset state.

999 **7 AllJoyn Translation**

1000 **7.1 Requirements Specific to an AllJoyn Translator**

1001 The translator shall be an AllJoyn Router Node. (This is a requirement so that users can expect
1002 that a certified OCF Bridge Device will be able to talk to any AllJoyn device, without the user having
1003 to buy some other device.)

1004 The requirements in this section apply when using algorithmic translation, and by default apply to
1005 deep translation unless the relevant specification for such deep translation specifies otherwise.

1006 **7.1.1 Exposing AllJoyn producer devices to OCF Clients**

1007 As specified in the OCF Security Specification, the value of the “di” property of OCF Devices
1008 (including Virtual OCF Devices) shall be established as part of Onboarding of that Virtual OCF
1009 Device.

1010
1011 Each AllJoyn object shall be mapped to one or more Virtual OCF Resources. If all AllJoyn
1012 interfaces can be translated to resource types on the same resource (as discussed below), there
1013 should be a single Virtual OCF Resource, and the path component of the URI of the Virtual OCF
1014 Resource shall be the AllJoyn object path, where each “_h” in the AllJoyn object path is
1015 transformed to “-” (hyphen), each “_d” in the AllJoyn object path is transformed to “.” (dot), each
1016 “_t” in the AllJoyn object path is transformed to “~” (tilde), and each “_u” in the AllJoyn object path
1017 is transformed to “_” (underscore). Otherwise, a Resource with that path shall exist with a
1018 Resource Type of [“oic.wk.col”, “oic.r.alljoynobject”] which is a Collection of links, where
1019 “oic.r.alljoynobject” is defined in Section 9.3, and the items in the collection are the Resources with
1020 the translated Resource Types as discussed below.

1021
1022 The value of the “piid” property of “/oic/d” for each Virtual OCF Device shall be the value of the
1023 OCF-defined AllJoyn field “org.openconnectivity.piid” in the AllJoyn About Announce signal, if that
1024 field exists, else it shall be calculated by the Translator as follows:

- 1025
- 1026 • If the AllJoyn device supports security, the value of the “piid” property value shall be the
1027 peer GUID.
- 1028 • If the AllJoyn device does not support security but the device is being bridged anyway (see
1029 section 9.2), the “piid” property value shall be derived from the Deviceld and Appld
1030 properties (in the About data), by concatenating the Deviceld value (not including any null
1031 termination) and the Appld bytes and using the result as the “name” to be used in the
1032 algorithm specified in IETF RFC 4122 section 4.3, with SHA-1 as the hash algorithm, and
1033 8f0e4e90-79e5-11e6-bdf4-0800200c9a66 as the name space ID. (This is to address the
1034 problem of being able to de-duplicate AllJoyn devices exposed via separate OCF Bridge
1035 Devices.)

1036 A translator implementation is encouraged to listen for AllJoyn About Announce signals matching
1037 any AllJoyn interface name. It can maintain a cache of information it received from these signals,
1038 and use the cache to quickly handle “/oic/res” queries from OCF Clients (without having to wait for
1039 Announce signals while handling the queries).

1040
1041 A translator implementation is encouraged to listen for other signals (including
1042 EmitsChangedSignal of properties) only when there is a client subscribed to a corresponding
1043 resource on a Virtual AllJoyn Device.

1044
1045 There are multiple types of AllJoyn interfaces, which shall be handled as follows.

- 1046 • If the AllJoyn interface is in a well-defined set (defined in OCF ASA Mapping or section
1047 7.1.1.1 below) of interfaces where standard forms exist on both the AllJoyn and OCF
1048 sides, the translator shall either:
 - 1049 a. follow the specification for translating that interface specially, or
 - 1050 b. not translate the AllJoyn interface.
- 1051 • If the AllJoyn interface is not in the well-defined set, the translator shall either:
 - 1052 a. not translate the AllJoyn interface, or
 - 1053 b. algorithmically map the AllJoyn interface as specified in section 7.2 to
1054 custom/vendor-defined Resource Types by converting the AllJoyn interface
1055 name to OCF resource type name(s).

1056

1057 An AllJoyn interface name shall be converted to a Device Type or a set of one or more OCF
 1058 Resource Types as follows:

- 1059 1) If the AllJoyn interface has any members, append a suffix “.<seeBelow>” where <seeBelow>
 1060 is described below.
- 1061 2) For each upper-case letter present in the entire string, replace it with a hyphen followed by
 1062 the lower-case version of that letter (e.g., convert “A” to “-a”).
- 1063 3) If an underscore appears followed by a (lower-case) letter or a hyphen, for each such
 1064 occurrence, replace the underscore with two hyphens (e.g., convert “_a” to “--a”, “_a” to
 1065 “---a”).
- 1066 4) For each underscore remaining, replace it with a hyphen (e.g., convert “_1” to “-1”).
- 1067 5) Prepend the “x.” prefix.

1068
 1069 Some examples are shown in the table below. The first three are normal AllJoyn names
 1070 converted to unusual OCF names. The last three are unusual AllJoyn names converted
 1071 (perhaps back) to normal OCF names. (“xn--” is a normal domain name prefix for the
 1072 Punycode-encoded form of an Internationalized Domain Name, and hence can appear in a
 1073 normal vendor-specific OCF name.)
 1074

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

1075
 1076 Each AllJoyn interface that has members and is using algorithmic mapping shall be mapped to one
 1077 or more Resource Types as follows:

- 1078 • AllJoyn Properties with the same EmitsChangedSignal value are mapped to the same
 1079 Resource Type where the value of the <seeBelow> label is the value of
 1080 EmitsChangedSignal. AllJoyn Properties with EmitsChangedSignal values of “const” or
 1081 “false”, are mapped to Resources that are not Observable, whereas AllJoyn Properties with
 1082 EmitsChangedSignal values of “true” or “invalidates” result in Resources that are
 1083 Observable. The Version property in an AllJoyn interface is always considered to have an
 1084 EmitsChangedSignal value of “const”, even if not specified in introspection XML. The name
 1085 of each property on the Resource Type shall be
 1086 “<ResourceType>.<AllJoynPropertyName>”, where each “_d” in the
 1087 <AllJoynPropertyName> is transformed to “.” (dot), and each “_h” in the
 1088 <AllJoynPropertyName> is transformed to “-” (hyphen).
- 1089 • Resource Types mapping AllJoyn Properties with access “readwrite” shall support the
 1090 “oic.if.rw” Interface. Resource Types mapping AllJoyn Properties with access “read” shall
 1091 support the “oic.if.r” Interface. Resource Types supporting both the “oic.if.rw” and “oic.if.r”
 1092 Interfaces shall choose “oic.if.r” as the default Interface.
- 1093 • Each AllJoyn Method is mapped to a separate Resource Type, where the value of the
 1094 <seeBelow> label is the AllJoyn Method name. The Resource Type shall support the
 1095 “oic.if.rw” Interface. Each argument of the AllJoyn Method shall be mapped to a separate
 1096 Property on the Resource Type, where the name of that Property is prefixed with
 1097 “<ResourceType>arg<#>”, where <#> is the 0-indexed position of the argument in the
 1098 AllJoyn introspection xml, in order to help get uniqueness across all Resource Types on
 1099 the same Resource. Therefore, when the AllJoyn argument name is not specified, the
 1100 name of that property is “<ResourceType>arg<#>”, where <#> is the 0-indexed position of
 1101 the argument in the AllJoyn introspection XML. In addition, that Resource Type has an
 1102 extra “<ResourceType>validity” property that indicates whether the rest of the properties

1103 have valid values. When the values are sent as part of an UPDATE response, the validity
1104 property is true, and any other properties have valid values. In a RETRIEVE (GET or
1105 equivalent in the relevant transport binding) response, the validity property is false, and
1106 any other properties can have meaningless values. If the validity property appears in an
1107 UPDATE request, its value shall be true (a value of false shall result in an error response).

- 1108 • Each AllJoyn Signal (whether sessionless, sessioncast, or unicast) is mapped to a separate
1109 Resource Type on an Observable Resource, where the value of the <seeBelow> label is
1110 the AllJoyn Signal name. The Resource Type shall support the "oic.if.r" Interface. Each
1111 argument of the AllJoyn Signal is mapped to a separate Property on the Resource Type,
1112 where the name of that Property is prefixed with "<ResourceType>arg<#>", where <#> is
1113 the 0-indexed position of the argument in the AllJoyn introspection xml, in order to help get
1114 uniqueness across all Resource Types on the same Resource. Therefore, when the
1115 AllJoyn argument name is not specified, the name of that property is
1116 "<ResourceType>arg<#>", where <#> is the 0-indexed position of the argument in the
1117 AllJoyn introspection XML. In addition, that Resource Type has an extra
1118 "<ResourceType>validity" property that indicates whether the rest of the properties have
1119 valid values. When the values are sent as part of a NOTIFY response, the validity property
1120 is true, and any other properties have valid values. In a RETRIEVE (GET or equivalent in
1121 the relevant transport binding) response, the validity property is false, and any other
1122 properties returned can have meaningless values. This is because in AllJoyn, the signals
1123 are instantaneous events, and the values are not necessarily meaningful beyond the
1124 lifetime of that message. Note that AllJoyn does have a TTL field that allows store-and-
1125 forward signals, but such support is not required in OCF 1.0. We expect that in the future,
1126 the TTL may be used to allow valid values in response to a RETRIEVE that is within the
1127 TTL.

1128 When an algorithmic mapping is used, AllJoyn data types shall be mapped to OCF property types
1129 according to Section 7.2.

1130

1131 If an AllJoyn operation fails, the translator shall send an appropriate OCF error response to the
1132 OCF client. If an AllJoyn error name is available and does not contain the
1133 "org.openconnectivity.Error.Code" prefix, it shall construct an appropriate OCF error message (e.g.,
1134 diagnostic payload if using CoAP) from the AllJoyn error name and AllJoyn error message (if any),
1135 using the form "<error name>: <error message>", with the <error name> taken from the AllJoyn
1136 error name field and the <error message> taken from the AllJoyn error message, and the CoAP
1137 error code set to an appropriate value (if CoAP is used). If an AllJoyn error name is available and
1138 contains the "org.openconnectivity.Error.Code" prefix, the OCF error message (e.g., diagnostic
1139 payload if using CoAP) should be taken from the AllJoyn error message (if any), and the CoAP
1140 error code (if CoAP is used) set to a value derived as follows; remove the
1141 "org.openconnectivity.Error.Code" prefix, and if the resulting error name is of the form "<#>" where
1142 <#> is an error code without a decimal (e.g., "404"), the CoAP error code shall be the error code
1143 indicated by the "<#>". Example: "org.openconnectivity.Error.Code404" becomes "404", which
1144 shall result in an error 4.04 for a CoAP transport.

1145 7.1.1.1 Exposing an AllJoyn producer application as a Virtual OCF Server

1146 Table 1 shows how OCF Device properties, as specified in Table 20 in the OCF 1.0 Core
1147 Specification, shall be derived, typically from fields specified in the AllJoyn About Interface
1148 Specification and AllJoyn Configuration Interface Specification.

1149

1150 If the AllJoyn About or Config data field has a mapping rule defined (as in Table 1, Table 2, Table
1151 3, and Table 4 below), the field name shall be translated based on that mapping rule; else if the
1152 AllJoyn About or Config data field has a fully qualified name (with a <domain> prefix (such as
1153 "com.example", "org.alljoyn"), the field name shall be translated based on the rules specified in

1154 Section 7.1.1 for mapping AllJoyn fields; else, the field shall not be translated as it may be incorrect
 1155 (error) or it has no valid mapping (such as daemonRealm and passCode).

1156
 1157

Table 1: oic.wk.d resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand ?	From AJ Field name	AJ Description	AJ Mand?
(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 the core specification this device is implemented to, The syntax is "core.major.minor"]	Y	(none)	Translator should return its own value	
Device ID	di	Unique identifier for Device. This value shall be as defined in [OCF Security] for DeviceID.	Y	(none)	Use as defined in the OCF Security Specification	
Protocol-Independent ID	piid	Unique identifier for OCF Device (UUID)	Y	org.openconnectivity.piid if it exists, else "Peer GUID" (not in About, but exposed by protocol) if authenticated, else Hash(DeviceId,Appld) where the Hash is done by concatenating the Device Id (not including any null terminator) and the Appld and using the algorithm in IETF RFC 4122 section 4.3, with SHA-1. 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.	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. DeviceId: Device identifier set by platform-specific means. Appld: A 128-bit globally unique identifier for the application. The Appld shall be a universally unique identifier as specified in IETF RFC 4122.	Peer GUID: conditionally Y DeviceId: Y Appld: Y
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	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 the OCF Core specification, additional values are	This specification 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	N, but required by IRB for all standard interfaces, and absence can be used to imply a constant (e.g., 0)

				formatted as "x.<interface name>.<Version property value>".	Version property may be determined through introspection alone. Note that AllJoyn specifies that the default value is 1 if the "org.gtk.GDBus.Since" annotation is absent.	
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

1158
1159
1160
1161
1162
1163
1164
1165
1166

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

Table 2 shows how OCF Device Configuration properties, as specified in Table 15 in the OCF 1.0 Core Specification, shall be derived:

Table 2: oic.wk.con resource type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
(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 specifies otherwise.	N	DefaultLanguage	The default language supported by the device. Specified as an IETF language tag listed in RFC 5646 .	Y

1167
1168
1169
1170
1171
1172
1173
1174
1175
1176

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

Table 3 shows how OCF Platform properties, as specified in Table 21 in the OCF 1.0 Core Specification, shall be derived, typically from fields specified in the AllJoyn About Interface Specification and AllJoyn Configuration Interface Specification.

Table 3: oic.wk.p Resource Type definition

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
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 section 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
-----------	-----	--	---	----------	--	---

1178
1179 Table 4 shows how OCF Platform Configuration properties, as specified in Table 16 in the OCF
1180 1.0 Core Specification, shall be derived:

1181
1182 **Table 4: oic.wk.con.p Resource Type definition**

To OCF Property title	OCF Property name	OCF Description	OCF Mand?	From AJ Field name	AJ Description	AJ Mand?
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.

1183
1184 In addition, the “oic.wk.mnt” properties Factory_Reset (“fr”) and Reboot (“rb”) shall be mapped to
1185 AllJoyn Configuration methods FactoryReset and Restart, respectively.

1186 **7.1.2 Exposing OCF resources to AllJoyn consumer applications**

1187 Unless specified otherwise, each OCF resource shall be mapped to a separate AllJoyn object.

1188
1189 Each OCF Server shall be exposed as a separate AllJoyn producer application, with its own About
1190 data. This allows platform-specific, device-specific, and resource-specific fields to all be
1191 preserved across translation. However, this requires that AllJoyn Claiming of such producer
1192 applications be solved in a way that does not require user interaction, but this is left as an
1193 implementation issue.

1194
1195 The AllJoyn producer application shall implement the “oic.d.virtual” AllJoyn interface. This allows
1196 translators to determine if a device is already being translated when multiple translators are
1197 present. The “oic.d.virtual” interface is defined as follows:

1198
1199 `<interface name="oic.d.virtual"/>`

1200
1201 The implementation may choose to implement this interface by the AllJoyn object at path “/oic/d”.

1202
1203 The AllJoyn peer ID shall be the OCF device ID (“di”).

1204
1205 Unless specified otherwise, the AllJoyn object path shall be the OCF URI path, where each “-”
1206 (hyphen) in the OCF URI path is transformed to “_h”, each “.” (dot) in the OCF URI path is
1207 transformed to “_d”, each “~” (tilde) in the OCF URI path is transformed to “_t”, and each “_”
1208 (underscore) in the OCF URI path is transformed to “_u”.

1209
1210 The AllJoyn About data shall be populated per Table 5 below.

1211

1212 A translator implementation is encouraged to maintain a cache of OCF resources to handle
 1213 Wholmplements queries from the AllJoyn side, and emit an Announce Signal for each OCF Server.
 1214 Specifically, the translator could always Observe "/oic/res" changes and only Observe other
 1215 resources when there is a client with a session on a Virtual AllJoyn Device.

1216 There are multiple types of resources, which shall be handled as follows.

- 1217 • If the Resource Type is in a well-defined set (defined in OCF ASA Mapping or section
 1218 7.1.2.1 below) of resource types where standard forms exist on both the AllJoyn and OCF
 1219 sides, the translator shall either:
 1220 a. follow the specification for translating that resource type specially, or
 1221 b. not translate the Resource Type.
- 1222 • If the Resource Type is not in the well-defined set (but is not a Device Type), the translator
 1223 shall either:
 1224 a. not translate the Resource Type, or
 1225 b. algorithmically map the Resource Type as specified in section 7.2 to a
 1226 custom/vendor-defined AllJoyn interface by converting the OCF Resource Type
 1227 name to an AllJoyn Interface name.
 1228
 1229

1230 An OCF Resource Type or Device Type shall be converted to an AllJoyn interface name as follows:

- 1231 1) Remove the "x." prefix if present
- 1232 2) For each occurrence of a hyphen (in order from left to right in the string):
 1233 a. If the hyphen is followed by a letter, replace both characters with a single upper-
 1234 case version of that letter (e.g., convert "-a" to "A").
 1235 b. Else, if the hyphen is followed by another hyphen followed by either a letter or a
 1236 hyphen, replace two hyphens with a single underscore (e.g., convert "--a" to "_a",
 1237 "---" to "_-").
 1238 c. Else, convert the hyphen to an underscore (i.e., convert "-" to "_").
 1239

1240 Some examples are shown in the table below. The first three are unusual OCF names
 1241 converted (perhaps back) to normal AllJoyn names. The last three are normal OCF names
 1242 converted to unusual AllJoyn names. ("xn--" is a normal domain name prefix for the Punycode-
 1243 encoded form of an Internationalized Domain Name, and hence can appear in a normal vendor-
 1244 specific OCF name.)
 1245

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

1246 An OCF Device Type is mapped to an AllJoyn interface with no members.
 1247
 1248

1249 Unless specified otherwise, each OCF Resource Type shall be mapped to an AllJoyn interface as
 1250 follows:

- 1251 • Each OCF property is mapped to an AllJoyn property in that interface, where each "." (dot)
 1252 in the OCF property is transformed to "_d", and each "-" (hyphen) in the OCF property is
 1253 transformed to "_h".
- 1254 • The EmitsChangedSignal value for each AllJoyn property shall be set to "true" if the
 1255 resource supports NOTIFY, or "false" if it does not. (The value is never set to "const" or
 1256 "invalidates" since those concepts cannot currently be expressed in OCF.)

- 1257 • The “access” attribute for each AllJoyn property shall be “read” if the OCF property is read-
1258 only, or “readwrite” if the OCF property is read-write.
- 1259 • If the resource supports DELETE, a Delete() method shall appear in the interface.
- 1260 • If the resource supports CREATE, a Create() method shall appear in the interface, with
1261 input arguments of each property of the resource to create. (Such information is not
1262 available algorithmically in OIC 1.1 but can be determined in OCF 1.0 via introspection.) If
1263 such information is not available, a CreateWithDefaultValues() method shall appear which
1264 takes no input arguments. In either case, the output argument shall be an OBJECT_PATH
1265 containing the path of the created resource.
- 1266 • If the resource supports UPDATE (i.e., the “oic.if.rw” or “oic.if,a” interface) then an AllJoyn
1267 property set operation (i.e., an org.freedesktop.DBus.Properties.Set() method call) shall be
1268 mapped to a Partial UPDATE (e.g., POST in CoAP) with the corresponding OCF property.
- 1269 • If a Resource has a Resource Type “oic.r.alljoynobject”, then instead of separately
1270 translating each of the Resources in the collection to its own AllJoyn object, all Resources
1271 in the collection shall instead be translated to a single AllJoyn object whose object path is
1272 the OCF URI path of the collection.

1273 OCF property types shall be mapped to AllJoyn data types according to Section 7.2.
1274

1275 If an OCF operation fails, the translator shall send an appropriate AllJoyn error response to the
1276 AllJoyn consumer. If an error message is present in the OCF response, and the error message
1277 (e.g., diagnostic payload if using CoAP) fits the pattern "<error name>: <error message>" where
1278 <error name> conforms to the AllJoyn error name syntax requirements, the AllJoyn error name
1279 and AllJoyn error message shall be extracted from the error message in the OCF response.
1280 Otherwise, the AllJoyn error name shall be "org.openconnectivity.Error.Code<#>" where <#> is the
1281 error code (e.g., CoAP error code) in the OCF response without a decimal (e.g., "404") and the
1282 AllJoyn error message is the error message in the OCF response.
1283

1284 7.1.2.1 Exposing an OCF server as a Virtual AllJoyn Producer

1285 The object description returned in the About interface shall be formed as specified in the AllJoyn
1286 About Interface Specification, and Table 5 shows how AllJoyn About Interface fields shall be
1287 derived, based on properties in “oic.wk.d”, “oic.wk.con”, “oic.wk.p”, and “oic.wk.con.p”.
1288

1289 **Table 5: AllJoyn About Data fields**

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
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.	N

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
					If absent, the translator shall return a constant, e.g., empty string	
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. For example, [{"language": "en", "value": "Dave's Laptop"}]	N
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 element, else return empty array	N
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	N

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
					containing the device description in the indicated language.	
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)		Translator 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, else field shall be absent)	Version of platform resident OS – string (defined by manufacturer)	N
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

1290
1291
1292

The AllJoyn field “org.openconnectivity.piid” shall be announced but shall not be localized and its D-Bus type signature shall be “s”. All other AllJoyn field names listed in Table 5 which have the

1293 prefix “org.openconnectivity.” shall be neither announced nor localized and their D-Bus type
 1294 signature shall be “s”.

1295
 1296 In addition, any additional vendor-defined properties in the OCF Device resource “/oic/d” (which
 1297 implements the “oic.wk.d” resource type) and the OCF Platform resource “/oic/p” (which
 1298 implements the “oic.wk.p” resource type) shall be mapped to vendor-defined fields in the AllJoyn
 1299 About data, with a field name formed by removing the leading “x.” from the property name.

1300
 1301 Table 6 shows how AllJoyn Configuration Interface fields shall be derived, based on properties in
 1302 “oic.wk.con” and “oic.wk.con.p”.

Table 6: AllJoyn Configuration Data fields

To AJ Field name	AJ Description	AJ Mand?	From OCF Property title	OCF Property name	OCF Description	OCF Mand?
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 string property as being in this language unless the property specifies otherwise.	N
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

1305
 1306 The AllJoyn field “org.openconnectivity.loc” shall be neither announced nor localized and its D-
 1307 Bus type signature shall be “ad”. All other AllJoyn field names listed in Table 5 which have the

1308 prefix “org.openconnectivity.” shall be neither announced nor localized and their D-Bus type
1309 signature shall be “s”.

1310
1311 In addition, the Configuration methods FactoryReset and Restart shall be mapped to “oic.wk.mnt”
1312 properties Factory_Reset (“fr”) and Reboot (“rb”), respectively, and any additional vendor-defined
1313 properties in the OCF Configuration resource (which implements the “oic.wk.con” resource type
1314 and optionally the “oic.wk.con.p” resource type) shall be mapped to vendor-defined fields the
1315 AllJoyn Configuration data, with a field name formed by removing the leading “x.” from the property
1316 name.

1317

1318 **7.2 On-the-Fly Translation from D-Bus and OCF payloads**

1319 The “dbus1” payload format is specified in the D-Bus Specification and AllJoyn adopted the D-Bus
1320 protocol and made it distributed over the network. The modifications done by AllJoyn to the format are all
1321 in the header part of the packet, not in the data payload itself, which remains compatible with “dbus1”. Other
1322 variants of the protocol that have been proposed by the Linux community (“GVariant” and “kdbus” payloads)
1323 contain slight incompatibilities and are not relevant for this discussion.

1324 **7.2.1 Translation without aid of introspection**

1325 This section describes how translators shall translate messages between the two payload formats in the
1326 absence of introspection metadata from the actual device. This situation arises in the following cases:

- 1327 • Requests to OIC 1.1 devices
- 1328 • Replies from OIC 1.1 devices
- 1329 • Content not described by introspection, such as the inner payload of AllJoyn properties of type
1330 “D-Bus VARIANT”.

1331 Since introspection is not available, the translator cannot know the rich JSON sub-type, only the underlying
1332 CBOR type and from that it can infer the JSON generic type, and hence translation is specified below in
1333 terms of those generic types.

1334 **7.2.1.1 Booleans**

1335 Boolean conversion is trivial since both sides support this type.

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

1336 **7.2.1.2 Numeric types**

1337 The translation of numeric types is lossy and that is unavoidable due to the limited expressiveness
1338 of the JSON generic types. This can only be solved with introspection.

1339 The translation of numeric types is direction-specific.

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) ⁽¹⁾	

From D-Bus type	To JSON type
"q" - INT16 (signed 16-bit)	
"" - INT32 (signed 32-bit)	
"x" - INT64 (signed 64-bit) ⁽¹⁾	
"d" - DOUBLE (IEEE 754 double precision)	

1340

From JSON type	To D-Bus type
number	"d" - DOUBLE ⁽²⁾

1341

1342 Notes and rationales:

1343 1. D-Bus payloads of types "t" (UINT64) and "x" (INT64) can contain values that cannot be perfectly
 1344 represented in IEEE 754 double-precision floating point. The RFCs governing JSON do not forbid
 1345 such numbers but caution that many implementations may not be able to deal with them.
 1346 Currently, OCF transports its payload using CBOR instead of JSON, which can represent those
 1347 numbers with fidelity. However, it should be noted that the OCF 1.0 Core Specification does
 1348 not allow for integral numbers outside the range $-2^{53} \leq x \leq 2^{53}$.

1349 2. To provide the most predictable result, all translations from OCF to AllJoyn produce values of
 1350 type "d" DOUBLE (IEEE 754 double precision).

1351 **7.2.1.3 Text strings**

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

1352

1353 Conversion between D-Bus and JSON strings is simple, as both require their content to be valid Unicode.
 1354 For example, an implementation can typically do a direct byte copy, as both protocols specify UTF-8 as
 1355 the encoding of the data, neither constrains the data to a given normalisation format nor specify whether
 1356 private-use characters or non-characters should be disallowed.

1357 Since the length of D-Bus strings is always known, it is recommended translators not use CBOR
 1358 indeterminate text strings (first byte 0x7f).

1359 **7.2.1.4 Byte arrays**

1360 The translation of a byte array is direction-specific.

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

1361

1362 The base64url encoding is specified in IETF RFC 4648 section 5.

1363 **7.2.1.5 D-Bus Variants**

D-Bus type	JSON type
"v" – VARIANT	<i>see below</i>

1364
 1365 D-Bus has a type called VARIANT ("v") that is a wrapper around any other D-Bus type. It's a way for the
 1366 type system to perform type-erasure. JSON, on the other hand, is not type-safe, which means that all
 1367 JSON values are, technically, variants. The conversion for a D-Bus variant to JSON is performed by
 1368 entering that variant and encoding the type carried inside as per the rules in this document.

1369 The algorithm must be recursive, as D-Bus variants are allowed to contain variants themselves.

1370 **7.2.1.6 D-Bus Object Paths and Signatures**

1371 The translation of D-Bus object paths and signatures is unidirectional (there is no mapping *to* them,
 1372 only *from* them). In the reverse direction, Section 7.2.1.3 always converts to D-Bus STRING
 1373 rather than OBJECT_PATH or SIGNATURE since it is assumed that "s" is the most common string
 1374 type in use.

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

1375
 1376 Both D-Bus object paths and D-Bus type signatures are US-ASCII strings with specific formation rules,
 1377 found in the D-Bus Specification. They are very seldom used and are not expected to be found in
 1378 resources subject to translation without the aid of introspection.

1379 **7.2.1.7 D-Bus Structures**

1380 The translation of the following types is direction-specific:

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

1381
 1382 D-Bus structures can be interpreted as a fixed-length array containing a pre-determined list of types for
 1383 each member. This is how such a structure is mapped to JSON: as an array of heterogeneous content,
 1384 which are the exact members of the D-Bus structure, in the order in which they appear in the structure.

1385 **7.2.1.8 Arrays**

1386 The translation of the following types is bidirectional:

D-Bus type	JSON type
"ay" - ARRAY of BYTE	(base64-encoded) string – see Section 7.2.1.4
"ae" - ARRAY of DICT_ENTRY	object – see Section 7.2.1.9

1387
 1388 The translation of the following types is direction-specific:

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

1389

1390

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

1391

1392 Aside from arrays of bytes and arrays of dictionary entries, which are mapped to JSON strings and objects
 1393 respectively, arrays in JSON cannot be constrained to a single type (i.e., heterogeneous arrays). For that
 1394 reason, strictly speaking all D-Bus arrays excepting arrays of bytes and arrays of dictionary entries must
 1395 first be converted to arrays of variant "av" and then that array can be converted to JSON.

1396 Conversion of D-Bus arrays of variants uses the conversion of variants as specified above, which simply
 1397 eliminates the distinction between a variant containing a given value and that value outside a variant. In
 1398 other words, the elements of a D-Bus array are extracted and sent as elements of the JSON array, as per
 1399 the other rules of this document.

1400 **7.2.1.9 Dictionaries / Objects**

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

1401

1402 The choice of "dictionary of STRING to VARIANT" is made because that is the most common type of
 1403 dictionary found in payloads and is an almost perfect superset of all possible dictionaries in D-Bus anyway.
 1404 Moreover, it can represent JSON Objects with fidelity, which is the representation that OCF uses in its data
 1405 models, which in turn means those D-Bus dictionaries will be able to carry with fidelity any OCF JSON
 1406 Object in current use.

1407 D-Bus dictionaries that are not mapping string to variant are first converted to those constraints and then
 1408 encoded in CBOR.

1409 **7.2.1.10 Non-translatable types**

D-Bus Type
"h" – UNIX_FD (Unix file descriptor)

JSON type
null
undefined (not officially valid JSON, but some implementations permit it)

1410

1411 The above types are not translatable, and the translator should drop the incoming message. None of the
 1412 types above are in current use by either AllJoyn, OIC 1.1, or future OCF 1.0 devices, so the inability to
 1413 translate them should not be a problem.

1414 **7.2.1.11 Examples**

1415

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"

Source D-Bus	JSON Result
VARIANT(INT32(0))	0
VARIANT(VARIANT(INT32(0)))	0
VARIANT(STRING("Hello"))	"Hello"

1416

1417

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"))} })} })

1418
1419 Note:
1420 1. This value cannot be represented with IEEE754 double-precision floating point without loss
1421 of information. It is also outside the currently-allowed range of integrals in OCF.
1422

1423 **7.2.2 Translation with aid of introspection**

1424 When introspection is available, the translator can use the extra metadata provided by the side offering the
1425 service to expose a higher-quality reply to the other side. This chapter details modifications to the translation
1426 described in the previous chapter when the metadata is found.

1427 Introspection metadata can be used for both translating requests to services and replies from those services.
1428 When used to translate requests, the introspection is “constraining”, since the translator must conform
1429 exactly to what that service expects. When used to translate replies, the introspection is “relaxing”, but may
1430 be used to inform the receiver what other possible values may be encountered in the future.

1431 Note that OCF introspection uses JSON types, media attributes, and format attributes, not CBOR encoding.
1432 The actual encoding of each JSON type is discussed in Section 12.3 of the OCF 1.0 Core Specification,
1433 JSON format attribute values are as defined in JSON Schema Validation, and JSON media attribute
1434 values are as defined in JSON Hyper-Schema.

1435 **7.2.2.1 Translation of the introspection itself**

1436 Note that both OCF 1.0 and AllJoyn require all services exposed to include introspection metadata, which
1437 means the translator will need to translate the introspection information on-the-fly for each OCF resource
1438 or AllJoyn producer it finds. The translator shall preserve as much of the original information as can be
1439 represented in the translated format. This includes both the information used in machine interactions and
1440 the information used in user interactions, such as description and documentation text.

1441 **7.2.2.2 Variability of introspection data**

1442 Introspection data is not a constant and the translator may find, upon discovering further services, that the
1443 D-Bus interface or OCF Resource Type it had previously encountered is different than previously seen. The
1444 translator needs to take care about how the destination side will react to a change in introspection.

1445 D-Bus interfaces used by AllJoyn services may be updated to newer versions, which means a given type
1446 of service may be offered by two distinct versions of the same interface. Updates to standardised interfaces
1447 must follow strict guidelines established by the AllSeen Interface Review Board, mapping each version to
1448 a different OCF Resource Type should be possible without much difficulty. However, there’s no guarantee
1449 that vendor-specific extensions follow those requirements. Indeed, there’s nothing preventing two revisions
1450 of a product to contain completely incompatible interfaces that have the same name and version number.

1451 On the opposite direction, the rules are much laxer. Since OCF specifies optional properties to its Resource
1452 Types, a simple monotonically-increasing version number like AllJoyn consumer applications expect is not
1453 possible.

1454 However, it should be noted that services created by the translator by “on-the-fly” translation will only be
1455 accessed by generic client applications. Dedicated applications will only use “deep binding” translation.

1456 **7.2.2.3 Numeric types**

1457 For numeric values, all D-Bus and JSON numeric types are treated equally as source and may all be
1458 translated into any of the other side’s types. When translating a request to a service, the translator need
1459 only verify whether there would be loss of information when translating from source to destination. For

1460 example, when translating the number 1.5 to either a JSON integer or to one of the D-Bus integral types,
 1461 there would be loss of information, in which case the translator should refuse the incoming message.
 1462 Similarly, the value 1,234,567 does not fit the range of a D-Bus byte, 16-bit signed or unsigned integer.

1463 When translating the reply from the service, the translator shall use the following rules.

1464 The following table indicates how to translate from a JSON type to the corresponding D-Bus type, where
 1465 the first matching row shall be used. If the JSON schema does not indicate the minimum value of a JSON
 1466 integer, 0 is the default. If the JSON schema does not indicate the maximum value of a JSON integer, 2^{32}
 1467 – 1 is the default. The resulting AllJoyn introspection XML shall contain “org.alljoyn.Bus.Type.Min” and
 1468 “org.alljoyn.Bus.Type.Max” annotations whenever the minimum or maximum, respectively, of the JSON
 1469 value is different from the natural minimum or maximum of the D-Bus type.

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 = “^0 ([1-9][0-9]{0,19})\$”	“t” (UINT64)
	pattern = “^0 (-?[1-9][0-9]{0,18})\$”	“x” (INT64)

1470

1471 The following table indicates how to translate from a D-Bus type to the corresponding JSON type.

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 “^0 ([1-9][0-9]{0,19})\$”.	IETF RFC 7159 section 6 explains that higher JSON integers are not interoperable.

From D-Bus type	To JSON type	Note
"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 " $^0 (-?[1-9][0-9]{0,18})\$\$$ ".	IETF RFC 7159 section 6 explains that other JSON integers are not interoperable.
"d" (double)	number	

1472

1473 7.2.2.4 Text string and byte arrays

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

1474

1475 There's no difference in the translation of text strings and byte arrays compared to the previous section.
 1476 This section simply lists the JSON equivalent types for the generated OCF introspection.

1477 In addition, the mapping of the following JSON Types is direction-specific:

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

1478

1479 JSON strings with any other format value (e.g., date-time, uri, etc.) or pattern value not shown in this table
 1480 above shall be treated the same as if the format and pattern attributes were absent, by simply mapping
 1481 the value to a D-Bus string.

1482 7.2.2.5 D-Bus Variants

D-Bus Type	JSON Type
"v" – VARIANT	<i>see below</i>

1483

1484 If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus VARIANT, the
 1485 translator should create such a variant and encode the incoming value as the variant's payload as per the
 1486 rules in the rest of this document.

1487 7.2.2.6 D-Bus Object Paths and Signatures

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

1488

1489 If the introspection of an AllJoyn producer indicates a value in a request should be a D-Bus Object Path or
1490 D-Bus Signature, the translator should perform a validity check in the incoming CBOR Text String. If the
1491 incoming data fails to pass this check, the message should be rejected.

1492 **7.2.2.7 D-Bus Structures**

1493 D-Bus structure members are described in the introspection XML with the
1494 "org.alljoyn.Bus.Struct.*StructureName*.Field.*fieldName*.Type" annotation. The translator shall use
1495 the AJSoftwareVersion field of the About data obtained from a bridged AllJoyn producer as follows.
1496 When the version of AllJoyn implemented on the Bridged Device is v16.10.00 or greater and the
1497 member annotations are present, the translator shall use a JSON object to represent a structure,
1498 mapping each member to the entry with that name. The translator needs to be aware that the
1499 incoming CBOR payload may have changed the order of the fields, when compared to the D-Bus
1500 structure. When the version of AllJoyn implemented on the Bridged Device is less than v16.10.00,
1501 the translator shall follow the rule for translating D-Bus structures without the aid of introspection
1502 data.

1503 **7.2.2.8 Arrays and Dictionaries**

1504 If the introspection of the AllJoyn interface indicates that the array is neither an ARRAY of BYTE ("ay") nor
1505 an ARRAY of VARIANT ("av") or that the dictionary is not mapping STRING to VARIANT ("a{sv}"), the
1506 translator shall apply the constraining or relaxing rules specified in other sections.

1507 Similarly, if the OCF introspection indicates a homogeneous array type, the information about the array's
1508 element type should be used as the D-Bus array type instead of VARIANT ("v").

1509 **7.2.2.9 Other JSON format attribute values**

1510 The JSON format attribute may include other custom attribute types. They are not known at this time, but
1511 it is expected that those types be handled by their type and representation alone.

1512 **7.2.2.10 Examples**

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})\$"

AllJoyn Source	AllJoyn Introspection Notes	Translated JSON Payload	OCF Introspection Notes
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" } }

1513

CBOR Payload	OCF Introspection Notes	Translated AllJoyn	AllJoyn Introspection Notes
0	"type": "integer"	INT32(0)	

CBOR Payload	OCF Introspection Notes	Translated AllJoyn	AllJoyn Introspection Notes
0	"type": "integer", "minimum": -2 ⁴⁰ , "maximum": 2 ⁴⁰	INT64(0)	org.alljoyn.Bus.Type.Min = -2 ⁴⁰ org.alljoyn.Bus.Type.Max = 2 ⁴⁰
0	"type": "integer", "minimum": 0, "maximum": 2 ⁴⁸	UINT64(0)	org.alljoyn.Bus.Type.Max = 2 ⁴⁸
0.0	"type": "number"	DOUBLE(0.0)	
[1]	JSON schema indicates: "type": "array", "items": { "type": "integer", "minimum": 0, "maximum": 2 ⁴⁶ }	ARRAY<UINT64>(1)	org.alljoyn.Bus.Type.Max = 2 ⁴⁶

1514 **8 Device Type Definitions**

1515 The required Resource Types are listed in the table below.

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

1516 **9 Resource Type definitions**

1517 **9.1 List of resource types**

1518 **Table 7 Alphabetical list of resource types**

Friendly Name (informative)	Resource Type (rt)	Section
Secure Mode	oic.r.securemode	9.2
AllJoyn Object	oic.r.alljoynobject	9.3

1519

1520 **9.2 Secure Mode**

1521 **9.2.1 Introduction**

1522 This resource describes a secure mode on/off feature (on/off).

1523 A secureMode value of "true" means that the feature is on, and:

- 1524 • any Bridged Server that cannot be communicated with securely shall not have a
- 1525 corresponding Virtual OCF Server, and

- any Bridged Client that cannot be communicated with securely shall not have a corresponding Virtual OCF Client.

A secureMode value of “false” means that the feature is off, and:

- any Bridged Server can have a corresponding Virtual OCF Server, and
- any Bridged Client can have a corresponding Virtual OCF Client.

9.2.2 Example URI Path

/example/SecureModeResURI

9.2.3 Resource Type

The resource type (rt) is defined as: oic.r.securemode.

9.2.4 RAML Definition

```
1536 #%RAML 0.8
1537 title: OCFSecureMode
1538 version: v1.0.0-20170531
1539 traits:
1540   - interface:
1541     queryParameters:
1542       if:
1543         enum: ["oic.if.rw", "oic.if.baseline"]
1544
1545 /example/SecureModeResURI:
1546   description: |
1547     This resource describes a secure mode on/off feature (on/off).
1548     A secureMode value of “true” means that the feature is on, and any Bridged Server that cannot
1549     be communicated with securely shall not have a corresponding Virtual OCF Server, and any Bridged
1550     Client that cannot be communicated with securely shall not have a corresponding Virtual OCF Client.
1551     A secureMode value of “false” means that the feature is off, any Bridged Server can have a
1552     corresponding Virtual OCF Server, and any Bridged Client can have a corresponding Virtual OCF
1553     Client.
1554
1555   is: ['interface']
1556
1557   get:
1558     description: |
1559       Retrieves the value of secureMode.
1560
1561     responses:
1562       200:
1563         body:
1564           application/json:
1565             schema: /
1566               {
1567                 "id": "https://www.openconnectivity.org/ocf-
1568                 apis/bridging/schemas/oic.r.securemode.json#",
1569                 "$schema": "http://json-schema.org/draft-04/schema#",
1570                 "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
1571                 reserved.",
1572                 "title": "Secure Mode",
1573                 "definitions": {
1574                   "oic.r.securemode": {
1575                     "type": "object",
1576                     "properties": {
1577                       "secureMode": {
1578                         "type": "boolean",
```

```

1578         "description": "Status of the Secure Mode"
1579     }
1580 }
1581 }
1582 },
1583 "type": "object",
1584 "allOf": [
1585     {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
1586     {"$ref": "#/definitions/oic.r.securemode"}
1587 ],
1588 "required": [ "secureMode" ]
1589 }
1590
1591 example: /
1592 {
1593     "rt":          ["oic.r.securemode"],
1594     "id":          "unique_example_id",
1595     "secureMode": false
1596 }
1597
1598 post:
1599     description: |
1600         Updates the value of secureMode.
1601     body:
1602         application/json:
1603             schema: /
1604             {
1605                 "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.securemode.json#",
1606                 "$schema": "http://json-schema.org/draft-04/schema#",
1607                 "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
reserved.",
1608                 "title": "Secure Mode",
1609                 "definitions": {
1610                     "oic.r.securemode": {
1611                         "type": "object",
1612                         "properties": {
1613                             "secureMode": {
1614                                 "type": "boolean",
1615                                 "description": "Status of the Secure Mode"
1616                             }
1617                         }
1618                     }
1619                 },
1620                 "type": "object",
1621                 "allOf": [
1622                     {"$ref": "../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
1623                     {"$ref": "#/definitions/oic.r.securemode"}
1624                 ],
1625                 "required": [ "secureMode" ]
1626             }
1627         }
1628     example: /
1629     {
1630         "id":          "unique_example_id",
1631         "secureMode": true
1632     }
1633
1634 responses:
1635     200:
1636         body:
1637             application/json:
1638                 schema: /
1639                 {
1640                     "id": "https://www.openconnectivity.org/ocf-
apis/bridging/schemas/oic.r.securemode.json#",

```

```

1641     "$schema": "http://json-schema.org/draft-04/schema#",
1642     "description": "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
1643 reserved.",
1644     "title": "Secure Mode",
1645     "definitions": {
1646       "oic.r.securemode": {
1647         "type": "object",
1648         "properties": {
1649           "secureMode": {
1650             "type": "boolean",
1651             "description": "Status of the Secure Mode"
1652           }
1653         }
1654       }
1655     },
1656     "type": "object",
1657     "allOf": [
1658       {"$ref": "../core/schemas/oic.core-schema.json#/definitions/oic.core"},
1659       {"$ref": "#/definitions/oic.r.securemode"}
1660     ],
1661     "required": [ "secureMode" ]
1662   }
1663   example: /
1664     {
1665       "id": "unique_example_id",
1666       "secureMode": true
1667     }
1668
1669

```

1670 9.2.5 Swagger2.0 Definition

```

1671 {
1672   "swagger": "2.0",
1673   "info": {
1674     "title": "OCFSecureMode",
1675     "version": "v1.0.0-20170531",
1676     "license": {
1677       "name": "copyright 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.",
1678       "x-description": "Redistribution and use in source and binary forms, with or without
1679 modification, are permitted provided that the following conditions are met:\n      1.
1680 Redistributions of source code must retain the above copyright notice, this list of conditions and
1681 the following disclaimer.\n      2. Redistributions in binary form must reproduce the above
1682 copyright notice, this list of conditions and the following disclaimer in the documentation and/or
1683 other materials provided with the distribution.\n\n      THIS SOFTWARE IS PROVIDED BY THE Open
1684 Connectivity Foundation, INC. \AS IS\ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
1685 LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR
1686 WARRANTIES OF NON-INFRINGEMENT, ARE DISCLAIMED.\n      IN NO EVENT SHALL THE Open Connectivity
1687 Foundation, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
1688 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
1689 OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n      HOWEVER CAUSED AND
1690 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
1691 OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
1692 OF SUCH DAMAGE.\n"
1693     }
1694   },
1695   "schemes": ["http"],
1696   "consumes": ["application/json"],
1697   "produces": ["application/json"],
1698   "paths": {
1699     "/example/SecureModeResURI" : {
1700       "get": {
1701         "description": "This resource describes a secure mode on/off feature (on/off).\n\na
1702 secureMode value of 'true' means that the feature is on, and any Bridged Server that cannot be
1703 communicated with securely shall not have a corresponding Virtual OCF Server, and any Bridged
1704 Client that cannot be communicated with securely shall not have a corresponding Virtual OCF
1705 Client.\n\na secureMode value of 'false' means that the feature is off, any Bridged Server can have a
1706 corresponding Virtual OCF Server, and any Bridged Client can have a corresponding Virtual OCF
1707 Client.\n\nRetrieves the value of secureMode.\n",
1708         "parameters": [
1709           {"$ref": "#/parameters/interface"}

```

```

1710     ],
1711     "responses": {
1712         "200": {
1713             "description": "",
1714             "x-example":
1715                 {
1716                     "rt": ["oic.r.securemode"],
1717                     "id": "unique_example_id",
1718                     "secureMode": false
1719                 }
1720             ,
1721             "schema": { "$ref": "#/definitions/SecureMode" }
1722         }
1723     },
1724 },
1725 "post": {
1726     "description": "Updates the value of secureMode.\n",
1727     "parameters": [
1728         { "$ref": "#/parameters/interface" },
1729         {
1730             "name": "body",
1731             "in": "body",
1732             "required": true,
1733             "schema": { "$ref": "#/definitions/SecureMode" },
1734             "x-example":
1735                 {
1736                     "id": "unique_example_id",
1737                     "secureMode": true
1738                 }
1739         }
1740     ],
1741     "responses": {
1742         "200": {
1743             "description": "",
1744             "x-example":
1745                 {
1746                     "id": "unique_example_id",
1747                     "secureMode": true
1748                 }
1749             ,
1750             "schema": { "$ref": "#/definitions/SecureMode" }
1751         }
1752     }
1753 },
1754 },
1755 },
1756 "parameters": {
1757     "interface": {
1758         "in": "query",
1759         "name": "if",
1760         "type": "string",
1761         "enum": ["oic.if.rw", "oic.if.baseline"]
1762     }
1763 },
1764 "definitions": {
1765     "SecureMode": {
1766         {
1767             "properties": {
1768                 "id": {
1769                     "description": "Instance ID of this specific resource",
1770                     "maxLength": 64,
1771                     "readOnly": true,
1772                     "type": "string"
1773                 },
1774                 "if": {
1775                     "description": "The interface set supported by this resource",
1776                     "items": {
1777                         "enum": [
1778                             "oic.if.baseline",
1779                             "oic.if.ll",
1780                             "oic.if.b",

```

```

1781         "oic.if.lb",
1782         "oic.if.rw",
1783         "oic.if.r",
1784         "oic.if.a",
1785         "oic.if.s"
1786     ],
1787     "type": "string"
1788 },
1789 "minItems": 1,
1790 "readOnly": true,
1791 "type": "array"
1792 },
1793 "n": {
1794     "description": "Friendly name of the resource",
1795     "maxLength": 64,
1796     "readOnly": true,
1797     "type": "string"
1798 },
1799 "rt": {
1800     "description": "Resource Type",
1801     "items": {
1802         "maxLength": 64,
1803         "type": "string"
1804     },
1805     "minItems": 1,
1806     "readOnly": true,
1807     "type": "array"
1808 },
1809 "secureMode": {
1810     "description": "Status of the Secure Mode",
1811     "type": "boolean"
1812 }
1813 },
1814 "required": [
1815     "secureMode"
1816 ],
1817 "type": "object"
1818 }
1819 }
1820 }
1821 }
1822
1823

```

1824 9.2.6 Property Definition

Property name	Value type	Mandatory	Access mode	Description
secureMode	boolean	Yes	Read Write	Status of the Secure Mode

1825 9.2.7 CRUDN behaviour

Example Resource URI	Create	Read	Update	Delete	Notify
/example/SecureModeResURI		get	post		get

1826

1827 9.3 AllJoyn Object

1828 9.3.1 Introduction

1829 This resource is a collection of resources that were all derived from the same AllJoyn object.

1830 9.3.2 Example URI Path

1831 /example/AllJoynObject/

1832 9.3.3 Resource Type

1833 The resource type (rt) is defined as: oic.r.alljoynobject.

```

1834 9.3.4 RAML Definition
1835 ##RAML 0.8
1836 title: OCFAllJoynObject
1837 version: v1.0.0-20170531
1838 traits:
1839 - interface-ll:
1840   queryParameters:
1841     if:
1842       enum: ["oic.if.ll"]
1843 - interface-baseline:
1844   queryParameters:
1845     if:
1846       enum: ["oic.if.baseline"]
1847 - interface-all:
1848   queryParameters:
1849     if:
1850       enum: ["oic.if.ll", "oic.if.baseline"]
1851
1852 /example/AllJoynObject/?if=oic.if.baseline:
1853   description: |
1854     This resource is a collection of resources that were all derived from the same AllJoyn object.
1855
1856   is: ['interface-baseline']
1857
1858   get:
1859     description: |
1860       Retrieves the current AllJoyn object information.
1861
1862     responses:
1863       200:
1864         body:
1865           application/json:
1866             schema: |
1867               {
1868                 "id": "https://www.openconnectivity.org/ocf-
1869                 apis/bridging/schemas/oic.r.alljoynobject.json#",
1870                 "$schema": "http://json-schema.org/draft-04/schema#",
1871                 "description": "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
1872                 reserved.",
1873                 "title": "AllJoyn Object",
1874                 "definitions": {
1875                   "oic.r.alljoynobject": {
1876                     "type": "object",
1877                     "allOf": [
1878                       {
1879                         "$ref": "../../core/schemas/oic.collection-
1880                         schema.json#/definitions/oic.collection"
1881                       },
1882                       {
1883                         "properties": {
1884                           "rt": {
1885                             "type": "array",
1886                             "minItems": 2,
1887                             "maxItems": 2,
1888                             "uniqueItems": true,
1889                             "items": {
1890                               "enum": ["oic.r.alljoynobject", "oic.wk.col"]
1891                             }
1892                           }
1893                         }
1894                     }
1895                   }
1896                 }
1897             }

```

```

1894         ]
1895     },
1896 },
1897 "type": "object",
1898 "allOf": [
1899     {"$ref": "../../../core/schemas/oic.core-schema.json#/definitions/oic.core"},
1900     {"$ref": "#/definitions/oic.r.alljoynobject"}
1901 ]
1902 }
1903
1904 example: /
1905 {
1906     "rt": ["oic.r.alljoynobject", "oic.wk.col"],
1907     "id": "unique_example_id",
1908     "links": [
1909         {"href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
1910 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:1111"}]},
1911         {"href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
1912 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:1111"}]},
1913         {"href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
1914 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:1111"}]},
1915         {"href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
1916 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:1111"}]}
1917     ]
1918 }
1919
1920 /example/AllJoynObject/?if=oic.if.ll:
1921
1922 description: |
1923     This resource is a collection of resources that were all derived from the same AllJoyn object.
1924
1925 is: ['interface-ll']
1926
1927 get:
1928
1929 description: |
1930     Retrieves the Links in the current AllJoyn object.
1931
1932 responses:
1933
1934 200:
1935
1936 body:
1937 application/json:
1938
1939 schema: /
1940 {
1941     "id": "https://www.openconnectivity.org/ocf-
1942 apis/bridging/schemas/oic.r.alljoynobject-ll#",
1943     "$schema": "http://json-schema.org/draft-04/schema#",
1944     "description": "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
1945 reserved.",
1946     "title": "AllJoyn Object Links List Schema",
1947     "definitions": {
1948         "oic.r.alljoynobject-ll": {
1949             "allOf": [
1950                 {
1951                     "$ref": "../../../core/schemas/oic.collection.linkslist-
1952 schema.json#/definitions/oic.collection.alllinks"
1953                 }
1954             ]
1955         }
1956     },
1957     "allOf": [
1958         {"$ref": "#/definitions/oic.r.alljoynobject-ll"}
1959     ]
1960 }
1961
1962 example: /
1963 [
1964     {"href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
1965 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:1111"}]},

```



```

1957         {"href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
1958 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
1959         {"href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
1960 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
1961         {"href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
1962 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]}
1963     ]
1964
1965

```

1966 9.3.5 Swagger2.0 Definition

```

1967 {
1968     "swagger": "2.0",
1969     "info": {
1970         "title": "OCFAllJoynObject",
1971         "version": "v1.0.0-20170531",
1972         "license": {
1973             "name": "copyright 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.",
1974             "x-description": "Redistribution and use in source and binary forms, with or without
1975 modification, are permitted provided that the following conditions are met:\n
1976 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
1977 the following disclaimer.\n
1978 2. Redistributions in binary form must reproduce the above
1979 copyright notice, this list of conditions and the following disclaimer in the documentation and/or
1980 other materials provided with the distribution.\n\n
1981 THIS SOFTWARE IS PROVIDED BY THE Open
1982 Connectivity Foundation, INC. \AS IS\ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
1983 LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR
1984 WARRANTIES OF NON-INFRINGEMENT, ARE DISCLAIMED.\n
1985 IN NO EVENT SHALL THE Open Connectivity
1986 Foundation, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
1987 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
1988 OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n
1989 HOWEVER CAUSED AND
1990 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
1991 OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
1992 OF SUCH DAMAGE.\n"
1993         }
1994     },
1995     "schemes": ["http"],
1996     "consumes": ["application/json"],
1997     "produces": ["application/json"],
1998     "paths": {
1999         "/example/AllJoynObject/?if=oic.if.ll" : {
2000             "get": {
2001                 "description": "This resource is a collection of resources that were all derived from the
2002 same AllJoyn object.\nRetrieves the Links in the current AllJoyn object.\n",
2003                 "parameters": [
2004                     {"$ref": "#/parameters/interface-ll"}
2005                 ],
2006                 "responses": {
2007                     "200": {
2008                         "description": "",
2009                         "x-example":
2010                         [
2011                             {"href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
2012 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
2013                             {"href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
2014 ["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
2015                             {"href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
2016 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]},
2017                             {"href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
2018 ["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]}
2019                         ]
2020                     },
2021                     "schema": {"$ref": "#/definitions/AllJoynObject-ll" }
2022                 }
2023             }
2024         },
2025         "/example/AllJoynObject/?if=oic.if.baseline" : {
2026             "get": {
2027                 "description": "This resource is a collection of resources that were all derived from the
2028 same AllJoyn object.\nRetrieves the current AllJoyn object information.\n",
2029                 "parameters": [

```

```

2027         {"$ref": "#/parameters/interface-baseline"}
2028     ],
2029     "responses": {
2030         "200": {
2031             "description": "",
2032             "x-example":
2033                 {
2034                     "rt": ["oic.r.alljoynobject", "oic.wk.col"],
2035                     "id": "unique_example_id",
2036                     "links": [
2037                         {
2038                             "href": "/myRes1URI", "rt": ["x.example.widget.false"], "if":
["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:1111"}]},
2039                             {
2040                                 "href": "/myRes2URI", "rt": ["x.example.widget.true"], "if":
["oic.if.r", "oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:1111"}]},
2041                                 {
2042                                     "href": "/myRes3URI", "rt": ["x.example.widget.method1"], "if":
["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:1111"}]},
2043                                     {
2044                                         "href": "/myRes4URI", "rt": ["x.example.widget.method2"], "if":
["oic.if.rw", "oic.if.baseline"], "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:1111"}]}
2045                                 ]
2046                             }
2047                         ]
2048                     },
2049                     "schema": {"$ref": "#/definitions/AllJoynObject" }
2050                 }
2051             }
2052         }
2053     },
2054     "parameters": {
2055         "interface-ll" : {
2056             "in" : "query",
2057             "name" : "if",
2058             "type" : "string",
2059             "enum" : ["oic.if.ll"]
2060         },
2061         "interface-baseline" : {
2062             "in" : "query",
2063             "name" : "if",
2064             "type" : "string",
2065             "enum" : ["oic.if.baseline"]
2066         },
2067         "interface-all" : {
2068             "in" : "query",
2069             "name" : "if",
2070             "type" : "string",
2071             "enum" : ["oic.if.ll", "oic.if.baseline"]
2072         }
2073     },
2074     "definitions": {
2075         "AllJoynObject-ll" :
2076             {
2077                 "allOf": [
2078                     {
2079                         "$ref": "../../core/schemas/oic.collection.linkslist-schema.json"
2080                     }
2081                 ]
2082             }
2083     },
2084     "AllJoynObject" :
2085         {
2086             "allOf": [
2087                 {
2088                     "$ref": "../../core/schemas/oic.collection-schema.json"
2089                 },
2090                 {
2091                     "properties": {
2092                         "id": {
2093                             "description": "Instance ID of this specific resource",
2094                             "maxLength": 64,
2095                             "readOnly": true,
2096                             "type": "string"
2097                         }
2098                     }
2099                 }
2100             ]
2101         }

```

```

2098     },
2099     "if": {
2100         "description": "The interface set supported by this resource",
2101         "items": {
2102             "enum": [
2103                 "oic.if.baseline",
2104                 "oic.if.ll",
2105                 "oic.if.b",
2106                 "oic.if.lb",
2107                 "oic.if.rw",
2108                 "oic.if.r",
2109                 "oic.if.a",
2110                 "oic.if.s"
2111             ],
2112             "type": "string"
2113         },
2114         "minItems": 1,
2115         "readOnly": true,
2116         "type": "array"
2117     },
2118     "n": {
2119         "description": "Friendly name of the resource",
2120         "maxLength": 64,
2121         "readOnly": true,
2122         "type": "string"
2123     },
2124     "rt": {
2125         "items": {
2126             "enum": [
2127                 "oic.r.alljoynobject",
2128                 "oic.wk.col"
2129             ]
2130         },
2131         "maxItems": 2,
2132         "minItems": 2,
2133         "type": "array",
2134         "uniqueItems": true
2135     }
2136 }
2137 }
2138 ],
2139 "type": "object"
2140 }
2141 }
2142 }
2143 }
2144 }

```

2145 **9.3.6 CRUDN behaviour**

Example Resource URI	Create	Read	Update	Delete	Notify
/example/AllJoynObject/?if=oic.if.baseline		get	post		get
/example/AllJoynObject/?if=oic.if.ll		get			get

2146