

OCF Bridging Specification

VERSION 2.2.1 | December 2020



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

Copyright Open Connectivity Foundation, Inc. © 2020.
All Rights Reserved.

CONTENTS

21 1 Scope..... 1

22 2 Normative references 1

23 3 Terms, definitions, and abbreviated terms 2

24 3.1 Terms and definitions..... 2

25 3.2 Abbreviated terms..... 5

26 4 Document conventions and organization..... 6

27 4.1 Conventions..... 6

28 4.2 Notation..... 6

29 5 Introduction 7

30 5.1 Translation between OCF and Non-OCF ecosystem - primitive concept of

31 Bridging 7

32 5.2 Bridge Platform..... 7

33 5.3 Symmetric vs. asymmetric bridging 8

34 5.4 General requirements 10

35 5.4.1 Requirements common to all Bridge Platforms..... 10

36 5.4.2 Requirements specific to Symmetric Bridge Platforms 10

37 5.5 VOD List 10

38 5.6 Resource discovery 10

39 5.7 "Deep translation" vs. "on-the-fly" 16

40 5.8 Security 16

41 6 Device type definitions 16

42 7 Resource type definitions 16

43 7.1 List of resource types..... 16

44 7.2 VOD List 17

45 7.2.1 Introduction 17

46 7.2.2 Example URI 17

47 7.2.3 Resource type 17

48 7.2.4 OpenAPI 2.0 definition..... 17

49 7.2.5 Property definition 19

50 7.2.6 CRUDN behaviour 19

53

Figures

54	Figure 1 – Server bridging to Non- OCF device.....	7
55	Figure 2 – Bridge Platform components	7
56	Figure 3 – Schematic overview of a Bridge Platform bridging non-OCF devices	8
57	Figure 4 – Asymmetric server bridge.....	9
58	Figure 5 – Asymmetric client bridge	9
59	Figure 6 – /oic/res example responses.....	16
60		

61

Tables

62	Table 1 – Device type definitions	16
63	Table 2 – Alphabetical list of resource types	17
64	Table 3 – The Property definitions of the Resource with type "rt" = "oic.r.vodlist".....	19
65	Table 4 – The CRUDN operations of the Resource with type "rt" = "oic.r.vodlist".	19
66		

67 1 Scope

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

73 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

116 OCF Resource to oneM2M Module Class Mapping, *Open Connectivity Foundation Resource to*
117 *oneM2M Module Class Mapping Specification*, version 2.0.2
118 Available at:
119 [https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specifi](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.0.2.pdf)
120 [cation_v2.0.2.pdf](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.0.2.pdf)
121 Latest version available at:
122 [https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specifi](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification.pdf)
123 [cation.pdf](https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification.pdf)

124 OCF Resource to Zigbee Cluster Mapping, *Open Connectivity Foundation Resource to Zigbee*
125 *Cluster Mapping Specification*, version 2.0.3
126 Available at:
127 [https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.](https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.0.3.pdf)
128 [0.3.pdf](https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_2.0.3.pdf)
129 Latest version available at:
130 https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification.pdf

131 Zigbee, *Zigbee Specification*, August 2015
132 <http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/>

133 Zigbee Cluster Library, *Zigbee Cluster Library Specification*, January 2016
134 <http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/>

135 **3 Terms, definitions, and abbreviated terms**

136 **3.1 Terms and definitions**

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

139 ISO and IEC maintain terminological databases for use in standardization at the following
140 addresses:

- 141 – ISO Online browsing platform: available at <https://www.iso.org/obp>
- 142 – IEC Electropedia: available at <http://www.electropedia.org/>

143 **3.1.1**

144 **Asymmetric Client Bridge**

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

148 **3.1.2**

149 **Asymmetric Server Bridge**

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

153 **3.1.3**
154 **Bridge**
155 OCF Device that has a Device Type of "oic.d.bridge", provides information on the set of Virtual
156 OCF Devices (3.1.24) that are resident on the same Bridge Platform.

157 **3.1.4**
158 **Bridge Platform**
159 Entity on which the Bridge (3.1.2) and Virtual OCF Devices (3.1.25) are resident

160 **3.1.5**
161 **Bridged Client**
162 logical entity that accesses data via a Bridged Protocol (3.1.5). For example, an AllJoyn Consumer
163 application is a Bridged Client

164 **3.1.6**
165 **Bridged Device**
166 Bridged Client (3.1.3) or Bridged Server (3.1.8).

167 **3.1.7**
168 **Bridged Protocol**
169 another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

170 **3.1.8**
171 **Bridged Resource**
172 represents an artefact modelled and exposed by a Bridged Protocol (3.1.5), for example an AllJoyn
173 object is a Bridged Resource.

174 **3.1.9**
175 **Bridged Resource Type**
176 schema used with a Bridged Protocol (3.1.5), for example AllJoyn Interfaces are Bridged Resource
177 Types.

178 **3.1.10 Bridged Server**
179 logical entity that provides data via a Bridged Protocol (3.1.5), for example an AllJoyn Producer is
180 a Bridged Server. More than one Bridged Server can exist on the same physical platform.

181 **3.1.11**
182 **Bridging Function**
183 Logic resident on the Bridge Platform (3.1.4) that performs that protocol mapping between OCF
184 and the Bridged Protocol (3.1.7); a Bridge Platform (3.1.4) may contain multiple Bridging Functions
185 dependent on the number of Bridged Protocols (3.1.7) supported.

186 **3.1.12**
187 **OCF Bridge Device**
188 OCF Device (3.1.11) that can represent devices that exist on the network but communicate using
189 a Bridged Protocol (3.1.5) rather than OCF protocols.

190 **3.1.13**
191 **OCF Client**
192 logical entity that accesses an OCF Resource (3.1.12) on an OCF Server (3.1.15), which might be
193 a Virtual OCF Server (3.1.26) exposed by the OCF Bridge Device (3.1.9)

194 **3.1.14**
195 **OCF Device**
196 logical entity that assumes one or more OCF roles (OCF Client (3.1.10), OCF Server (3.1.15)). More
197 than one OCF Device can exist on the same physical platform.

198 **3.1.15**
199 **OCF Resource**
200 represents an artefact modelled and exposed by the OCF Framework

201 **3.1.16**
202 **OCF Resource Property**
203 significant aspect or notion including metadata that is exposed through the OCF Resource (3.1.12)

204 **3.1.17**
205 **OCF Resource Type**
206 OCF Resource Property (3.1.13) that represents the data type definition for the OCF Resource
207 (3.1.12)

208 **3.1.18**
209 **OCF Server**
210 logical entity with the role of providing resource state information and allowing remote control of its
211 resources

212 **3.1.19**
213 **oneM2M Application**
214 In an OCF-oneM2M asymmetric bridge environment, the oneM2M application represents the
215 oneM2M control point (i.e. client) being mapped to a virtual OCF client.

216 **3.1.20**
217 **Symmetric, Asymmetric Bridging**
218 in symmetric bridging, a bridge device exposes OCF Server(s) (3.1.15) to another ecosystem and
219 exposes other ecosystem's server(s) to OCF. In asymmetric bridging, a bridge device exposes
220 OCF Server(s) (3.1.15) to another ecosystem or exposes another ecosystem's server(s) to OCF,
221 but not both.

222 **3.1.21**
223 **Virtual Bridged Client**
224 logical representation of an OCF Client (3.1.10), which an OCF Bridge Device (3.1.9) exposes to
225 Bridged Servers (3.1.8).

226 **3.1.22**
227 **Virtual Bridged Server**
228 logical representation of an OCF Server (3.1.15), which an OCF Bridge Device (3.1.9) exposes to
229 Bridged Clients (3.1.3).

230 **3.1.23**
231 **Virtual OCF Client**
232 logical representation of a Bridged Client (3.1.3), which an OCF Bridge Device (3.1.9) exposes to
233 OCF Servers (3.1.15)

234 **3.1.24**
235 **Virtual OCF Device**
236 Virtual OCF Client (3.1.23) or Virtual OCF Server (3.1.26).

237 **3.1.25**
238 **Virtual OCF Resource**
239 logical representation of a Bridged Resource (3.1.6), which an OCF Bridge Device (3.1.9) exposes
240 to OCF Clients (3.1.10)

241 **3.1.26**
242 **Virtual OCF Server**
243 logical representation of a Bridged Server (3.1.8), which an OCF Bridge Device (3.1.9) exposes to
244 OCF Clients (3.1.10).

245 **3.1.27**
246 **Zigbee Attribute**
247 data entity which represents a physical quantity or state within Zigbee. This data is communicated
248 to other devices using commands.

249 **3.1.28**
250 **Zigbee Cluster**
251 one or more Zigbee Attributes (3.1.27), commands, behaviours, and dependencies, which supports
252 an independent utility or application function. The term may also be used for an implementation or
253 instance of such on an endpoint.

254 **3.1.29**
255 **Zigbee Server**
256 cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint.
257 Typically this interface supports all or most of the attributes of the cluster. A server cluster
258 communicates with a corresponding remote client cluster with the same identifier.

259 **3.1.30**
260 **Zigbee 3.0 Server**
261 Zigbee Server (3.1.29) which is built on Zigbee 3.0 stack

262 **3.1.31**
263 **Zigbee Client**
264 cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint.
265 Typically this interface sends commands that manipulate the attributes on the corresponding
266 Zigbee Server (3.1.29). A client cluster communicates with a corresponding remote server cluster
267 with the same identifier.

268 **3.1.32**
269 **Zigbee 3.0 Client**
270 Zigbee Client (3.1.31) which is built on Zigbee 3.0 stack

271 **3.1.33**
272 **Zigbee Device**
273 unique device identifier and a set of mandatory and optional clusters to be implemented on a single
274 Zigbee endpoint. The term may also be used for an implementation or instance on an endpoint. In
275 this document, the unique identifier of a Zigbee Device maps to an OCF Device Type.

276 **3.1.34**
277 **Zigbee 3.0 Device**
278 Zigbee Device (3.1.33) which is built on Zigbee 3.0 stack

279 **3.2 Abbreviated terms**

280 **3.2.1**
281 **CRUDN**
282 Create, Read, Update, Delete, and Notify

283 **3.2.2**
284 **CSV**
285 Comma separated value

286 4 Document conventions and organization

287 4.1 Conventions

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

292 4.2 Notation

293 In this document, features are described as required, recommended, allowed or DEPRECATED as
294 follows:

295 Required (or shall or mandatory).

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

299 Recommended (or should).

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

306 Allowed (or allowed).

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

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

312 Conditionally required (CR)

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

316 DEPRECATED

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

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

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

325 **5 Introduction**

326 **5.1 Translation between OCF and Non-OCF ecosystem - primitive concept of Bridging**

327 The details of Bridging may be implemented in many ways, for example, by using a Bridge Platform
 328 with an entity handler to interface directly to a Non-OCF device as shown in Figure 1.



329
 330 **Figure 1 – Server bridging to Non- OCF device**

331 On start-up the Bridge Platform runs the entity handlers which discover the non-OCF systems (e.g.,
 332 Heart Rate Sensor Device) and create Resources for each Device or functionality discovered. The
 333 entity handler creates a Resource for each discovered Device or functionality and binds itself to
 334 that Resource. These Resources are made discoverable by the Bridge Platform.

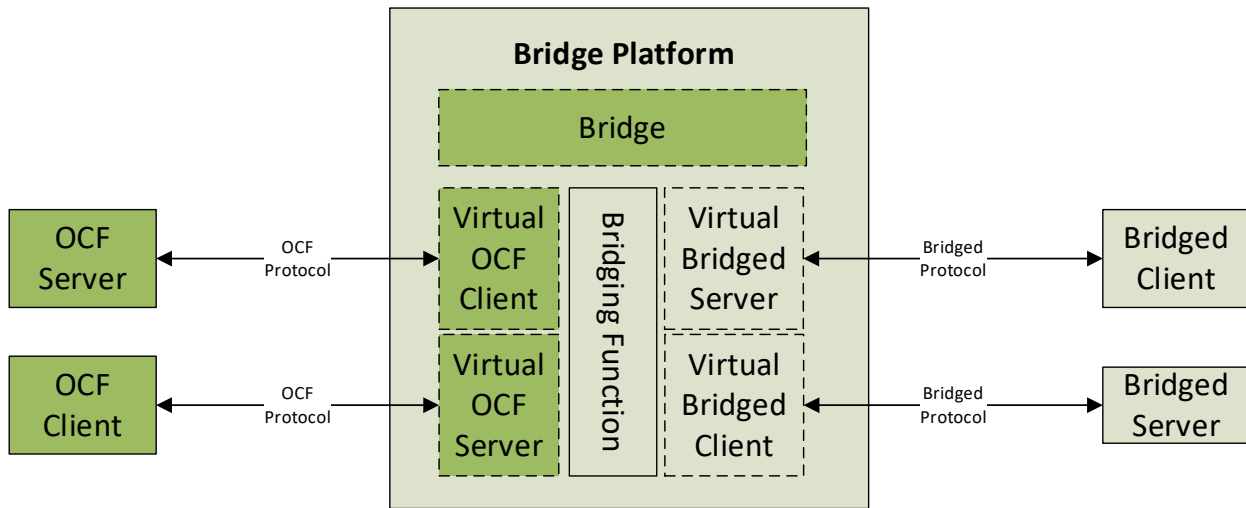
335 Once the Resources are created and made discoverable, then the Client Device can discover these
 336 Resources and operate on them using the mechanisms described in ISO/IEC 30118-1:2018. The
 337 requests to a Resource on the Bridge Platform are then interpreted by the entity handler and
 338 forwarded to the non-OCF device using the protocol supported by the non-OCF device. The
 339 returned information from the non-OCF device is then mapped to the appropriate response for that
 340 Resource.

341 Current OCF Bridging architecture implements the entity handler in the form of VOD.

342

343 **5.2 Bridge Platform**

344 This clause describes the functionality of a Bridge Platform; such a device is illustrated in Figure 2.

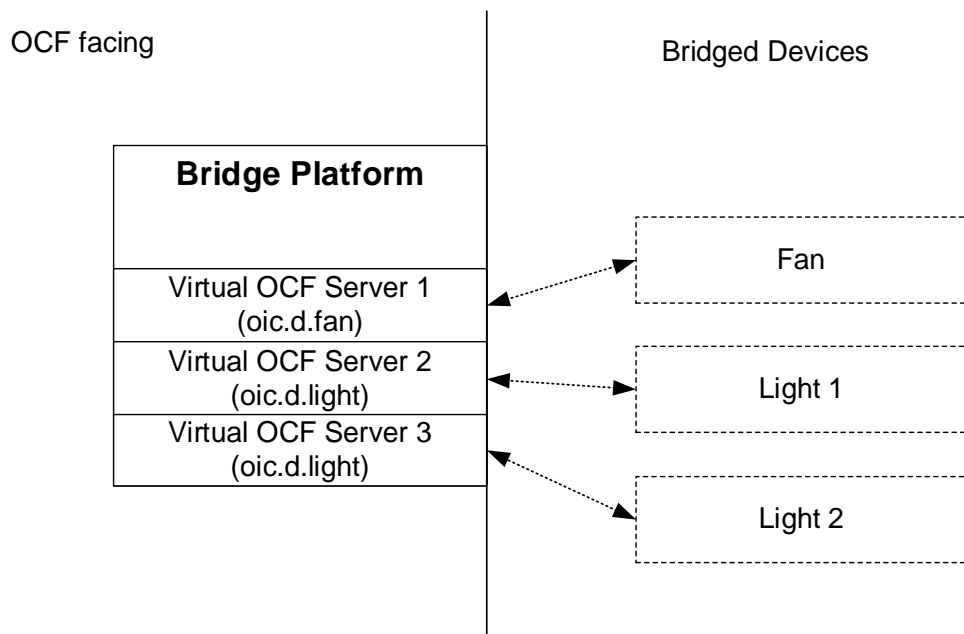


345

346 **Figure 2 – Bridge Platform components**

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

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



362

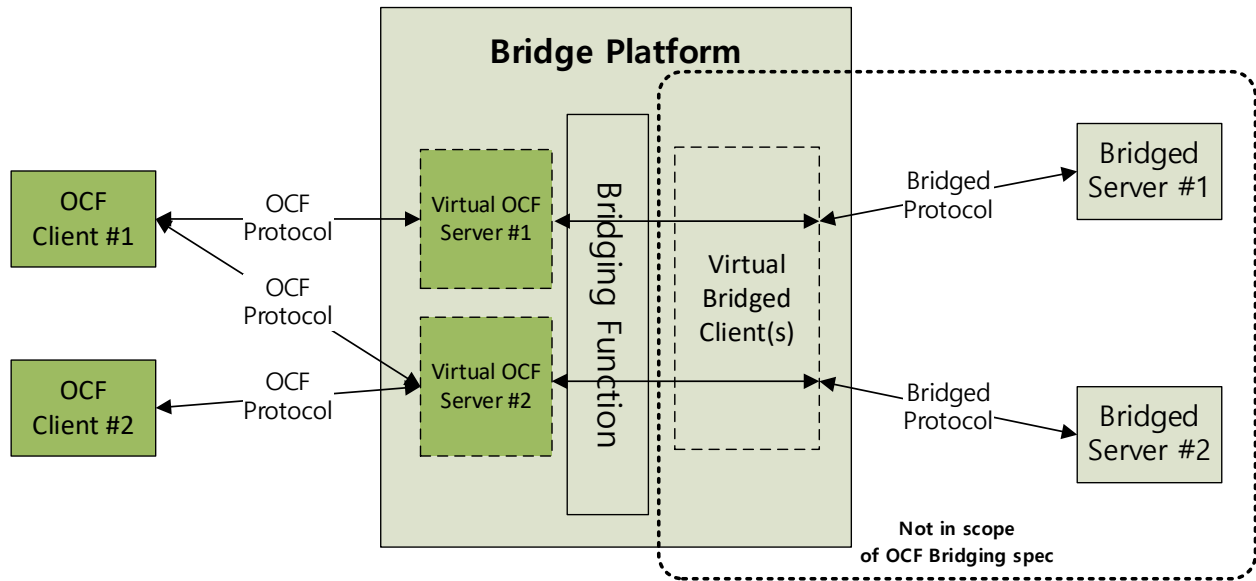
363 **Figure 3 – Schematic overview of a Bridge Platform bridging non-OCF devices**

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

368 **5.3 Symmetric vs. asymmetric bridging**

369 There are two kinds of bridging: Symmetric, Asymmetric. In symmetric bridging, a bridge device
 370 exposes OCF server(s) to another ecosystem and exposes other ecosystem's server(s) to OCF. In
 371 asymmetric bridging, a bridge device exposes OCF server(s) to another ecosystem or exposes
 372 another ecosystem's server(s) to OCF, but not both. The former case is called an Asymmetric
 373 Server Bridge (see Figure 4), the latter case is called an Asymmetric Client Bridge (see Figure 5)

374



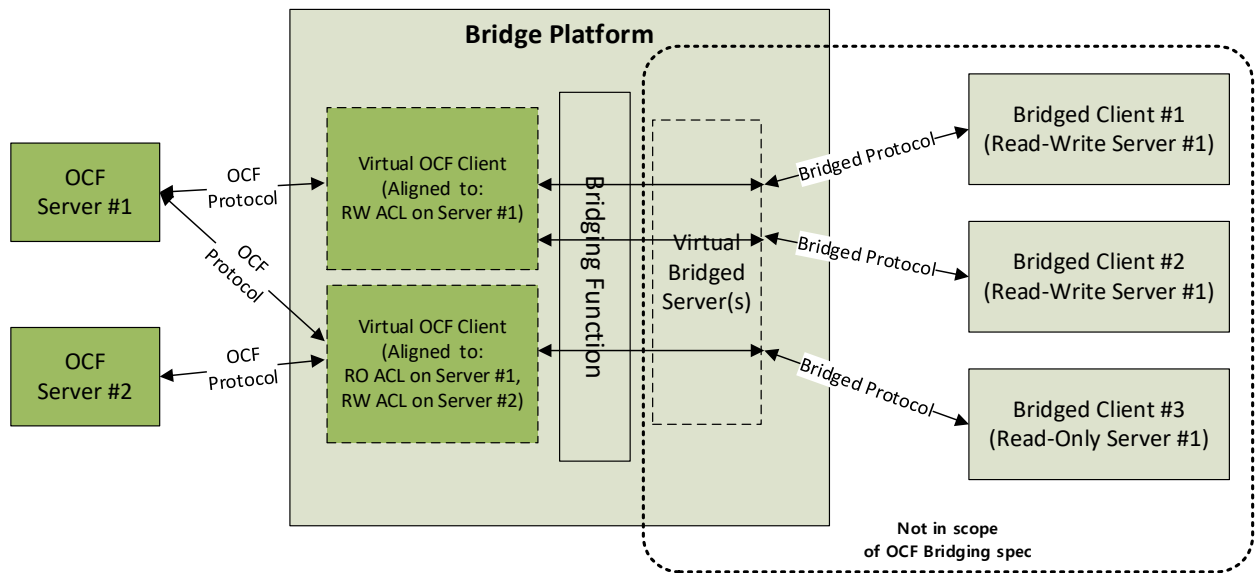
375

376

Figure 4 – Asymmetric server bridge

377 In Figure 4 each Bridged Server is exposed as a Virtual OCF Server to OCF side. These Virtual
 378 OCF Servers are same as normal OCF Servers except that they have additional rt value
 379 ("oic.d.virtual") for "/oic/d". The details of the Virtual Bridged Client are not in scope of this
 380 document.

381



382

383

Figure 5 – Asymmetric client bridge

384 Figure 5 shows that each access to the OCF Server is modelled as a Virtual OCF Client. Those
 385 accesses can be aggregated if their target OCF servers and access permissions are same,
 386 therefore a Virtual OCF Client can tackle multiple Bridged Clients.

387 **5.4 General requirements**

388 **5.4.1 Requirements common to all Bridge Platforms**

389 A VOD shall have a Device Type that contains "oic.d.virtual". This allows Bridge Platforms to
390 determine if a device is already being translated when multiple Bridge Platforms are present or
391 Clients to determine if corresponding Server is a VOD or not.

392 Each Bridged Device shall be exposed as a separate Virtual OCF Server or Client, with its own
393 OCF Endpoint, and set of mandatory Resources (as defined in ISO/IEC 30118-1:2018 and ISO/IEC
394 30118-2:2018).

395 Discovery of a VOD is the same as for an ordinary OCF Device; that is the VOD shall respond to
396 multicast discovery requests. This allows platform-specific, device-specific, and resource-specific
397 fields to all be preserved across translation.

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

401 **5.4.2 Requirements specific to Symmetric Bridge Platforms**

402 In addition to the requirements mentioned in 5.4.1, Symmetric Bridging shall satisfy following
403 requirements.

404 The Bridge Platform shall check the protocol-independent UUID ("piid" in OCF) of each device and
405 shall not advertise back into a Bridged Protocol a device originally seen via that Bridged Protocol.
406 The Bridge Platform shall stop translating any Bridged Protocol device exposed in OCF via another
407 Bridge Platform if the Bridge Platform sees the device via the Bridged Protocol. Similarly, the Bridge
408 Platform shall not advertise an OCF Device back into OCF, and the Bridge Platform shall stop
409 translating any OCF device exposed in the Bridged Protocol via another Bridge Platform if the
410 Bridge Platform sees the device via OCF. These require that the Bridge Platform can determine
411 when a device is already being translated. A VOD shall be indicated on the OCF Security Domain
412 with a Device Type of "oic.d.virtual". How a Bridge Platform determines if a device is already being
413 translated on a non-OCF Security Domain is described in the protocol-specific clauses (e.g. clause
414 1).

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

418 **5.5 VOD List**

419 For maintenance purposes, the Bridge maintains a list of VODs. This list includes Virtual OCF
420 Servers and Virtual OCF Clients created by the Bridge Platform and subsequently on-boarded, as
421 specified in ISO/IEC 30118-2:2018. A single instance of the Resource Type that defines the VOD
422 list (see clause 7.2) shall be exposed by the Bridge. Please refer to ISO/IEC 30118-2:2018 for
423 detailed operational requirements for the VOD list.

424 **5.6 Resource discovery**

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

431 A Bridge Platform and any exposed VODs shall each respond to network discovery commands.
432 The response to a RETRIEVE on "/oic/res" shall only include the devices that match the RETRIEVE
433 request.

434 For example, if a Bridge exposes VODs for the fan and lights shown in Figure 3, and an OCF Client
435 performs a discovery request with a content format of "application/vnd.ocf+cbor", there will be four
436 discrete responses, one for the Bridge, one for the virtual fan Device, and two for the virtual light
437 Devices. Note that what is returned is not in the JSON format but in a suitable encoding as defined
438 in ISO/IEC 30118-1:2018.

439 Response from the Bridge:

```
440 [
441   {
442     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
443     "href": "/oic/res",
444     "rel": "self",
445     "rt": ["oic.wk.res"],
446     "if": ["oic.if.ll", "oic.if.baseline"],
447     "p": {"bm": 3},
448     "eps": [{"ep": "coap://[2001:db8:a::b1d4]:55555"},
449             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
450   },
451   {
452     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
453     "href": "/oic/d",
454     "rt": ["oic.wk.d", "oic.d.bridge"],
455     "if": ["oic.if.r", "oic.if.baseline"],
456     "p": {"bm": 3},
457     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
458   },
459   {
460     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
461     "href": "/oic/p",
462     "rt": ["oic.wk.p"],
463     "if": ["oic.if.r", "oic.if.baseline"],
464     "p": {"bm": 3},
465     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
466   },
467   {
468     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
469     "href": "/oic/sec/doxm",
470     "rt": ["oic.r.doxm"],
471     "if": ["oic.if.baseline"],
472     "p": {"bm": 1},
473     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
474   },
475   {
476     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
477     "href": "/oic/sec/pstat",
478     "rt": ["oic.r.pstat"],
479     "if": ["oic.if.baseline"],
480     "p": {"bm": 1},
481     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
482   },
483   {
484     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
485     "href": "/oic/sec/cred",
486     "rt": ["oic.r.cred"],
487     "if": ["oic.if.baseline"],
488     "p": {"bm": 1},
489     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
490   }
491 ]
```



```

490     },
491     {
492         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
493         "href": "/oic/sec/acl2",
494         "rt": ["oic.r.acl2"],
495         "if": ["oic.if.baseline"],
496         "p": {"bm": 1},
497         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
498     },
499     {
500         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
501         "href": "/myIntrospection",
502         "rt": ["oic.wk.introspection"],
503         "if": ["oic.if.r", "oic.if.baseline"],
504         "p": {"bm": 3},
505         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
506     },
507     {
508         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
509         "href": "/myVodlist",
510         "rt": ["oic.r.vodlist"],
511         "if": ["oic.if.r", "oic.if.baseline"],
512         "p": {"bm": 3},
513         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:11111"}]
514     }
515 ]
516
517 Response from the Fan VOD:
518 [
519     {
520         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
521         "href": "/oic/res",
522         "rt": ["oic.wk.res"],
523         "if": ["oic.if.ll", "oic.if.baseline"],
524         "p": {"bm": 3},
525         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
526     },
527     {
528         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
529         "href": "/oic/d",
530         "rt": ["oic.wk.d", "oic.d.fan", "oic.d.virtual"],
531         "if": ["oic.if.r", "oic.if.baseline"],
532         "p": {"bm": 3},
533         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
534     },
535     {
536         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
537         "href": "/oic/p",
538         "rt": ["oic.wk.p"],
539         "if": ["oic.if.r", "oic.if.baseline"],
540         "p": {"bm": 3},
541         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
542     },
543     {
544         "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
545         "href": "/myFan",
546         "rt": ["oic.r.switch.binary"],
547         "if": ["oic.if.a", "oic.if.baseline"],
548         "p": {"bm": 3},
549         "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
550     },
551     {

```

```

552     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
553     "href": "/oic/sec/doxm",
554     "rt": ["oic.r.doxm"],
555     "if": ["oic.if.baseline"],
556     "p": {"bm": 1},
557     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
558 },
559 {
560     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
561     "href": "/oic/sec/pstat",
562     "rt": ["oic.r.pstat"],
563     "if": ["oic.if.baseline"],
564     "p": {"bm": 1},
565     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
566 },
567 {
568     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
569     "href": "/oic/sec/cred",
570     "rt": ["oic.r.cred"],
571     "if": ["oic.if.baseline"],
572     "p": {"bm": 1},
573     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
574 },
575 {
576     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
577     "href": "/oic/sec/acl2",
578     "rt": ["oic.r.acl2"],
579     "if": ["oic.if.baseline"],
580     "p": {"bm": 1},
581     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
582 },
583 {
584     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
585     "href": "/myFanIntrospection",
586     "rt": ["oic.wk.introspection"],
587     "if": ["oic.if.r", "oic.if.baseline"],
588     "p": {"bm": 3},
589     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:22222"}]
590 }
591 ]
592
593 Response from the first Light VOD:
594 [
595 {
596     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
597     "href": "/oic/res",
598     "rt": ["oic.wk.res"],
599     "if": ["oic.if.ll", "oic.if.baseline"],
600     "p": {"bm": 3},
601     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
602 },
603 {
604     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
605     "href": "/oic/d",
606     "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
607     "if": ["oic.if.r", "oic.if.baseline"],
608     "p": {"bm": 3},
609     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:33333"}]
610 },
611 {
612     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
613     "href": "/oic/p",

```

```

614     "rt": ["oic.wk.p"],
615     "if": ["oic.if.r", "oic.if.baseline"],
616     "p": {"bm": 3},
617     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
618   },
619   {
620     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
621     "href": "/myLight",
622     "rt": ["oic.r.switch.binary"],
623     "if": ["oic.if.a", "oic.if.baseline"],
624     "p": {"bm": 3},
625     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
626   },
627   {
628     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
629     "href": "/oic/sec/doxm",
630     "rt": ["oic.r.doxm"],
631     "if": ["oic.if.baseline"],
632     "p": {"bm": 1},
633     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
634   },
635   {
636     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
637     "href": "/oic/sec/pstat",
638     "rt": ["oic.r.pstat"],
639     "if": ["oic.if.baseline"],
640     "p": {"bm": 1},
641     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
642   },
643   {
644     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
645     "href": "/oic/sec/cred",
646     "rt": ["oic.r.cred"],
647     "if": ["oic.if.baseline"],
648     "p": {"bm": 1},
649     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
650   },
651   {
652     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
653     "href": "/oic/sec/acl2",
654     "rt": ["oic.r.acl2"],
655     "if": ["oic.if.baseline"],
656     "p": {"bm": 1},
657     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
658   },
659   {
660     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
661     "href": "/myLightIntrospection",
662     "rt": ["oic.wk.introspection"],
663     "if": ["oic.if.r", "oic.if.baseline"],
664     "p": {"bm": 3},
665     "eps": [{"ep": "coaps://[2001:db8:a:b1d4]:33333"}]
666   }
667 ]
668
669 Response from the second Light VOD:
670 [
671   {
672     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
673     "href": "/oic/res",
674     "rt": ["oic.wk.res"],
675     "if": ["oic.if.ll", "oic.if.baseline"],

```

```

676     "p": {"bm": 3},
677     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
678   },
679   {
680     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
681     "href": "/oic/d",
682     "rt": ["oic.wk.d", "oic.d.light", "oic.d.virtual"],
683     "if": ["oic.if.r", "oic.if.baseline"],
684     "p": {"bm": 3},
685     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
686   },
687   {
688     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
689     "href": "/oic/p",
690     "rt": ["oic.wk.p"],
691     "if": ["oic.if.r", "oic.if.baseline"],
692     "p": {"bm": 3},
693     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
694   },
695   {
696     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
697     "href": "/myLight",
698     "rt": ["oic.r.switch.binary"],
699     "if": ["oic.if.a", "oic.if.baseline"],
700     "p": {"bm": 3},
701     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
702   },
703   {
704     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
705     "href": "/oic/sec/doxm",
706     "rt": ["oic.r.doxm"],
707     "if": ["oic.if.baseline"],
708     "p": {"bm": 1},
709     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
710   },
711   {
712     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
713     "href": "/oic/sec/pstat",
714     "rt": ["oic.r.pstat"],
715     "if": ["oic.if.baseline"],
716     "p": {"bm": 1},
717     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
718   },
719   {
720     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
721     "href": "/oic/sec/cred",
722     "rt": ["oic.r.cred"],
723     "if": ["oic.if.baseline"],
724     "p": {"bm": 1},
725     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
726   },
727   {
728     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
729     "href": "/oic/sec/acl2",
730     "rt": ["oic.r.acl2"],
731     "if": ["oic.if.baseline"],
732     "p": {"bm": 1},
733     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
734   },
735   {
736     "anchor": "ocf://8202138e-aa22-452c-b512-9ebad02bef7c",
737     "href": "/myLightIntrospection",

```

```

738     "rt": ["oic.wk.introspection"],
739     "if": ["oic.if.r", "oic.if.baseline"],
740     "p": {"bm": 3},
741     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:44444"}]
742   }
743 ]

```

744 **Figure 6 – /oic/res example responses**

745 **5.7 "Deep translation" vs. "on-the-fly"**

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

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

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

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

765 **5.8 Security**

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

769 **6 Device type definitions**

770 The required Resource Types are listed in Table 1.

771 **Table 1 – Device type definitions**

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

772 **7 Resource type definitions**

773 **7.1 List of resource types**

774 Table 2 lists the Resource Types defined in this document.

Table 2 – Alphabetical list of resource types

Friendly Name (informative)	Resource Type (rt)	Clause
VOD List	oic.r.vodlist	10.4

776

777 **7.2 VOD List**778 **7.2.1 Introduction**

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

780 **7.2.2 Example URI**

781 /VODListResURI

782 **7.2.3 Resource type**

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

784 **7.2.4 OpenAPI 2.0 definition**

```

785 {
786   "swagger": "2.0",
787   "info": {
788     "title": "VOD List",
789     "version": "2019-05-16",
790     "license": {
791       "name": "OCF Data Model License",
792       "url":
793         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
794         CENSE.md",
795       "x-copyright": "Copyright 2019 Open Connectivity Foundation, Inc. All rights reserved."
796     },
797     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
798   },
799   "schemes": ["http"],
800   "consumes": ["application/json"],
801   "produces": ["application/json"],
802   "paths": {
803     "/VODListResURI" : {
804       "get": {
805         "description": "This Resource describes the VODs that have been onboarded on the Bridge
806         Platform.\n",
807         "parameters": [
808           {"$ref": "#/parameters/interface"}
809         ],
810         "responses": {
811           "200": {
812             "description" : "Example response payload",
813             "x-example":
814             {
815               "rt": ["oic.r.vodlist"],
816               "vods": [
817                 {
818                   "n": "Smoke sensor",
819                   "di": "54919CA5-4101-4AE4-595B-353C51AA1234",
820                   "econame": "Z-Wave"
821                 },
822                 {
823                   "n": "Thermostat",
824                   "di": "54919CA5-4101-4AE4-595B-353C51AA5678",
825                   "econame": "Zigbee"
826                 }
827               ]
828             },
829           "schema": { "$ref": "#/definitions/vodlist" }
830         }
831       }
832     }

```

```

831     }
832   }
833 }
834 },
835 "parameters": {
836   "interface": {
837     "in": "query",
838     "name": "if",
839     "type": "string",
840     "enum": ["oic.if.r", "oic.if.baseline"]
841   }
842 },
843 "definitions": {
844   "vodentry": {
845     "description": "Information for a VOD created by the Bridge",
846     "type": "object",
847     "properties": {
848       "n": {
849         "$ref":
850 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
851 schema.json#/definitions/n"
852       },
853       "di": {
854         "$ref": "https://openconnectivityfoundation.github.io/core/schemas/oic.types-
855 schema.json#/definitions/uuid"
856       },
857       "econame": {
858         "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD",
859         "type": "string",
860         "enum": [ "BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave" ],
861         "readOnly": true
862       }
863     },
864     "required": ["n", "di", "econame"]
865   },
866   "vodlist": {
867     "type": "object",
868     "properties": {
869       "n": {
870         "$ref":
871 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
872 schema.json#/definitions/n"
873       },
874       "rt": {
875         "description": "Resource Type",
876         "items": {
877           "maxLength": 64,
878           "type": "string",
879           "enum": ["oic.r.vodlist"]
880         },
881         "minItems": 1,
882         "uniqueItems": true,
883         "readOnly": true,
884         "type": "array"
885       },
886       "id": {
887         "$ref":
888 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
889 schema.json#/definitions/id"
890       },
891       "if": {
892         "description": "The OCF Interface set supported by this Resource",
893         "items": {
894           "enum": [
895             "oic.if.baseline",
896             "oic.if.r"
897           ],
898           "type": "string"
899         },
900         "minItems": 2,

```

```

901     "uniqueItems": true,
902     "readOnly": true,
903     "type": "array"
904   },
905   "vods": {
906     "description": "Array of information per VOD created by the Bridge",
907     "type": "array",
908     "minItems": 0,
909     "uniqueItems": true,
910     "readOnly": true,
911     "items": {
912       "$ref": "#/definitions/vodentry"
913     }
914   }
915 },
916 "required": ["vods"]
917 }
918 }
919 }
920

```

921 7.2.5 Property definition

922 Table 3 defines the Properties that are part of the "oic.r.vodlist" Resource Type.

923 **Table 3 – The Property definitions of the Resource with type "rt" = "oic.r.vodlist".**

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

924 7.2.6 CRUDN behaviour

925 Table 4 defines the CRUDN operations that are supported on the "oic.r.vodlist" Resource Type.

926 **Table 4 – The CRUDN operations of the Resource with type "rt" = "oic.r.vodlist".**

Create	Read	Update	Delete	Notify
	get			observe

927

