

OCF Core Specification

VERSION 2.1.0 | November 2019



OPEN CONNECTIVITY
FOUNDATION™

CONTACT admin@openconnectivity.org

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

Legal Disclaimer

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

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

17 Copyright © 2016-2019 Open Connectivity Foundation, Inc. All rights reserved.

18 Copying or other form of reproduction and/or distribution of these works are strictly prohibited.
19

CONTENTS

20			
21	1	Scope	1
22	2	Normative references	1
23	3	Terms, definitions, and abbreviated terms	3
24	3.1	Terms and definitions.....	3
25	3.2	Abbreviated terms.....	7
26	4	Document conventions and organization.....	8
27	4.1	Conventions.....	8
28	4.2	Notation.....	8
29	4.3	Data types	9
30	5	Architecture	11
31	5.1	Overview	11
32	5.2	Principle	11
33	5.3	Functional block diagram	12
34	5.4	Framework.....	14
35	6	Identification and addressing	14
36	6.1	Introduction.....	14
37	6.2	Identification	15
38	6.2.1	Device and Platform identification.....	15
39	6.2.2	Resource identification and addressing	15
40	6.3	Namespace:.....	16
41	6.4	Network addressing	16
42	7	Resource model	17
43	7.1	Introduction.....	17
44	7.2	Resource	18
45	7.3	Property.....	18
46	7.3.1	Introduction	18
47	7.3.2	Common Properties	19
48	7.4	Resource Type	20
49	7.4.1	Introduction	20
50	7.4.2	Resource Type Property	21
51	7.4.3	Resource Type definition	21
52	7.4.4	Multi-value "rt" Resource	23
53	7.5	Device Type.....	23
54	7.6	OCF Interface	24
55	7.6.1	Introduction	24
56	7.6.2	OCF Interface Property.....	24
57	7.6.3	OCF Interface methods.....	25
58	7.7	Resource representation.....	43
59	7.8	Structure.....	43
60	7.8.1	Introduction	43
61	7.8.2	Resource relationships (Links).....	43
62	7.8.3	Collections.....	49

63	7.8.4	Atomic Measurement	51
64	7.9	Query Parameters	53
65	7.9.1	Introduction	53
66	7.9.2	Use of multiple parameters within a query	53
67	7.9.3	Application to multi-value "rt" Resources	53
68	7.9.4	OCF Interface specific considerations for queries	54
69	8	CRUDN	54
70	8.1	Overview	54
71	8.2	CREATE	55
72	8.2.1	Overview	55
73	8.2.2	CREATE request	55
74	8.2.3	Processing by the Server	55
75	8.2.4	CREATE response	56
76	8.3	RETRIEVE	56
77	8.3.1	Overview	56
78	8.3.2	RETRIEVE request	56
79	8.3.3	Processing by the Server	56
80	8.3.4	RETRIEVE response	56
81	8.4	UPDATE	57
82	8.4.1	Overview	57
83	8.4.2	UPDATE request	57
84	8.4.3	Processing by the Server	57
85	8.4.4	UPDATE response	58
86	8.5	DELETE	58
87	8.5.1	Overview	58
88	8.5.2	DELETE request	58
89	8.5.3	Processing by the Server	59
90	8.5.4	DELETE response	59
91	8.6	NOTIFY	59
92	8.6.1	Overview	59
93	8.6.2	NOTIFICATION response	59
94	9	Network and connectivity	59
95	9.1	Introduction	59
96	9.2	Architecture	59
97	9.3	IPv6 network layer requirements	61
98	9.3.1	Introduction	61
99	9.3.2	IPv6 node requirements	61
100	10	OCF Endpoint	61
101	10.1	OCF Endpoint definition	61
102	10.2	OCF Endpoint information	62
103	10.2.1	Introduction	62
104	10.2.2	"ep"	62
105	10.2.3	"pri"	63
106	10.2.4	OCF Endpoint information in "eps" Parameter	63

107	10.3	OCF Endpoint discovery	64
108	10.3.1	Introduction	64
109	10.3.2	Implicit discovery	64
110	10.3.3	Explicit discovery with "/oic/res" response	64
111	11	Functional interactions	66
112	11.1	Introduction.....	66
113	11.2	Resource discovery	67
114	11.2.1	Introduction	67
115	11.2.2	Resource based discovery: mechanisms	67
116	11.2.3	Resource based discovery: Finding information	68
117	11.2.4	Resource discovery using "/oic/res"	74
118	11.2.5	Multicast discovery using "/oic/res"	75
119	11.3	Notification	76
120	11.3.1	Overview	76
121	11.3.2	Observe.....	76
122	11.4	Introspection.....	77
123	11.4.1	Overview	77
124	11.4.2	Usage of Introspection.....	80
125	11.5	Semantic Tags.....	81
126	11.5.1	Introduction	81
127	11.5.2	Semantic Tag definitions	82
128	12	Messaging.....	84
129	12.1	Introduction.....	84
130	12.2	Mapping of CRUDN to CoAP.....	84
131	12.2.1	Overview	84
132	12.2.2	URIs	84
133	12.2.3	CoAP method with request and response	85
134	12.2.4	Content-Format negotiation	86
135	12.2.5	OCF-Content-Format-Version information.....	87
136	12.2.6	Content-Format policy	88
137	12.2.7	CRUDN to CoAP response codes	88
138	12.2.8	CoAP block transfer.....	89
139	12.2.9	Generic requirements for CoAP multicast	89
140	12.3	Mapping of CRUDN to CoAP serialization over TCP	90
141	12.3.1	Overview	90
142	12.3.2	URIs.....	90
143	12.3.3	CoAP method with request and response	90
144	12.3.4	Content-Format negotiation	90
145	12.3.5	OCF-Content-Format-Version information.....	90
146	12.3.6	Content-Format policy	90
147	12.3.7	CRUDN to CoAP response codes	90
148	12.3.8	CoAP block transfer.....	90
149	12.3.9	Keep alive (connection health).....	90
150	12.4	Payload Encoding in CBOR	90

151	13 Security	91
152	Annex A (normative) Resource Type definitions	92
153	A.1 List of Resource Type definitions	92
154	A.2 Atomic Measurement links list representation	92
155	A.2.1 Introduction	92
156	A.2.2 Example URI	92
157	A.2.3 Resource type	92
158	A.2.4 OpenAPI 2.0 definition.....	92
159	A.2.5 Property definition	99
160	A.2.6 CRUDN behaviour	100
161	A.3 Collection.....	100
162	A.3.1 Introduction	100
163	A.3.2 Example URI	100
164	A.3.3 Resource type	100
165	A.3.4 OpenAPI 2.0 definition.....	100
166	A.3.5 Property definition	108
167	A.3.6 CRUDN behaviour	109
168	A.4 Device	109
169	A.4.1 Introduction	109
170	A.4.2 Well-known URI	109
171	A.4.3 Resource type	109
172	A.4.4 OpenAPI 2.0 definition.....	109
173	A.4.5 Property definition	112
174	A.4.6 CRUDN behaviour	113
175	A.5 Introspection Resource	114
176	A.5.1 Introduction	114
177	A.5.2 Well-known URI	114
178	A.5.3 Resource type	114
179	A.5.4 OpenAPI 2.0 definition.....	114
180	A.5.5 Property definition	116
181	A.5.6 CRUDN behaviour	116
182	A.6 Platform	117
183	A.6.1 Introduction	117
184	A.6.2 Well-known URI	117
185	A.6.3 Resource type	117
186	A.6.4 OpenAPI 2.0 definition.....	117
187	A.6.5 Property definition	120
188	A.6.6 CRUDN behaviour	120
189	A.7 Discoverable Resources	121
190	A.7.1 Introduction	121
191	A.7.2 Well-known URI	121
192	A.7.3 Resource type	121
193	A.7.4 OpenAPI 2.0 definition.....	121
194	A.7.5 Property definition	125

195	A.7.6	CRUDN behaviour	126
196		Annex B (informative) OpenAPI 2.0 Schema Extension.....	127
197	B.1	OpenAPI 2.0 Schema Reference.....	127
198	B.2	OpenAPI 2.0 Introspection empty file	127
199		Annex C (normative) Semantic Tag enumeration support.....	128
200	C.1	Introduction.....	128
201	C.2	"tag-pos-desc" supported enumeration.....	128
202		Bibliography.....	129
203			
204			

205
206
207

Figures

208	Figure 1 – Architecture - concepts	12
209	Figure 2 – Functional block diagram	13
210	Figure 3 – Communication layering model	14
211	Figure 4 – Example Resource	18
212	Figure 5 – CREATE operation.....	55
213	Figure 6 – RETRIEVE operation	56
214	Figure 7 – UPDATE operation.....	57
215	Figure 8 – DELETE operation	58
216	Figure 9 – High Level Network & Connectivity Architecture	60
217	Figure 10 – Resource based discovery: Finding information.....	68
218	Figure 11 – Observe Mechanism.....	76
219	Figure 12 – Example usage of oneOf JSON schema	79
220	Figure 13 – Interactions to check Introspection support and download the Introspection	
221	Device Data.	81
222	Figure 14 – "tag-pos-rel" definition.....	83
223	Figure 15 – Content-Format Policy for backward compatible OCF Clients negotiating lower	
224	OCF Content-Format-Version	88
225	Figure C.1 – Enumeration for "tag-pos-desc" Semantic Tag.....	128
226	Figure C.2 – Definition of "tag-pos-desc" Semantic Tag values	128

227

Tables

228

229

230	Table 1 – Additional OCF Types	10
231	Table 2 – Name Property Definition	20
232	Table 3 – Resource Identity Property Definition	20
233	Table 4 – Resource Type Common Property definition.....	21
234	Table 5 – Example foobar Resource Type.....	22
235	Table 6 – Example foobar Properties	22
236	Table 7 – Resource Interface Property definition.....	24
237	Table 8 – OCF standard OCF Interfaces	25
238	Table 9 – Batch OCF Interface Example	32
239	Table 10 – Link target attributes list	45
240	Table 11 – "bm" Property definition.....	46
241	Table 12 – Resource Types Property definition	48
242	Table 13 – Mandatory Resource Types Property definition.....	48
243	Table 14 – Common Properties for Collections (in addition to Common Properties defined	
244	in 7.3.2)	50

245	Table 15 – Common Properties for Atomic Measurement (in addition to Common	
246	Properties defined in 7.3.2)	51
247	Table 16 – Atomic Measurement Resource Type	52
248	Table 17 – Properties for Atomic Measurement (in addition to Common Properties defined	
249	in 7.3.2)	53
250	Table 18 – Parameters of CRUDN messages	54
251	Table 19 – "ep" value for Transport Protocol Suite	63
252	Table 20 – List of Core Resources	67
253	Table 21 – Mandatory discovery Core Resources	69
254	Table 22 – "oic.wk.res" Resource Type definition	69
255	Table 23 – Protocol scheme registry	70
256	Table 24 – "oic.wk.d" Resource Type definition	71
257	Table 25 – "oic.wk.p" Resource Type definition	73
258	Table 26 – Introspection Resource	80
259	Table 27 – "oic.wk.introspection" Resource Type definition	80
260	Table 28 – "tag-pos-desc" Semantic Tag definition	82
261	Table 29 – "tag-pos-rel" Semantic Tag definition	83
262	Table 30 – "tag-func-desc" Semantic Tag definition	84
263	Table 31 – CoAP request and response	85
264	Table 32 – OCF Content-Formats	86
265	Table 33 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option	
266	Numbers	87
267	Table 34 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version	
268	Representation	87
269	Table 35 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-	
270	Version Representation	87
271	Table A.1 – Alphabetized list of Core Resources	92
272	Table A.2 – The Property definitions of the Resource with type "rt" =	
273	"oic.wk.atomicmeasurement".	99
274	Table A.3 – The CRUDN operations of the Resource with type "rt" =	
275	"oic.wk.atomicmeasurement".	100
276	Table A.4 – The Property definitions of the Resource with type "rt" = "oic.wk.col".	108
277	Table A.5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col".	109
278	Table A.6 – The Property definitions of the Resource with type "rt" = "oic.wk.d".	113
279	Table A.7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d".	113
280	Table A.8 – The Property definitions of the Resource with type "rt" =	
281	"oic.wk.introspection".	116
282	Table A.9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".	117
283	Table A.10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".	120
284	Table A.11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p".	120
285	Table A.12 – The Property definitions of the Resource with type "rt" = "None".	125

286 Table A.13 – The CRUDN operations of the Resource with type "rt" = "None"..... 126
287
288

289 **1 Scope**

290 The OCF Core specifications are divided into a set of documents:

- 291 – Core specification (this document): The Core specification document specifies the Framework,
292 i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles
293 implementation for Internet of Things (IoT) usages and ecosystems. This document is
294 mandatory for all Devices to implement.
- 295 – Core optional specification: The Core optional specification document specifies the Framework,
296 i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles
297 implementation for Internet of Things (IoT) usages and ecosystems that can optionally be
298 implemented by any Device.
- 299 – Core extension specification(s): The Core extension specification(s) document(s) specifies
300 optional OCF Core functionality that are significant in scope (e.g., Wi-Fi easy setup, Cloud).

301 **2 Normative references**

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

305 ISO 8601, *Data elements and interchange formats – Information interchange –Representation of*
306 *dates and times*, International Standards Organization, December 3, 2004

307 ISO/IEC DIS 20924, *Information Technology – Internet of Things – Vocabulary*, June 2018
308 <https://www.iso.org/standard/69470.html>

309 ISO/IEC 30118-2:2018, *Information technology – Open Connectivity Foundation (OCF)*
310 *Specification – Part 2: Security specification*
311 <https://www.iso.org/standard/74239.html>
312 Latest version available at: https://openconnectivity.org/specs/OCF_Security_Specification.pdf

313 IETF RFC 768, *User Datagram Protocol*, August 1980
314 <https://www.rfc-editor.org/info/rfc768>

315 IETF RFC 3339, *Date and Time on the Internet: Timestamps*, July 2002
316 <https://www.rfc-editor.org/info/rfc3339>

317 IETF RFC 3986, *Uniform Resource Identifier (URI): General Syntax*, January 2005.
318 <https://www.rfc-editor.org/info/rfc3986>

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

321 IETF RFC 4287, *The Atom Syndication Format*, December 2005,
322 <https://www.rfc-editor.org/info/rfc4287>

323 IETF RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, September
324 2007
325 <https://www.rfc-editor.org/info/rfc4941>

326 IETF RFC 5646, *Tags for Identifying Languages*, September 2009
327 <https://www.rfc-editor.org/info/rfc5646>

328 IETF RFC 6347, *Datagram Transport Layer Security Version 1.2*, January 2012
329 <https://www.rfc-editor.org/info/rfc6347>

330 IETF RFC 6434, *IPv6 Node Requirements*, December 2011
331 <https://www.rfc-editor.org/info/rfc6434>

332 IETF RFC 6573, *The Item and Collection Link Relations*, April 2012
333 <https://www.rfc-editor.org/info/rfc6573>

334 IETF RFC 6690, *Constrained RESTful Environments (CoRE) Link Format*, August 2012
335 <https://www.rfc-editor.org/info/rfc6690>

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

338 IETF RFC 7084, *Basic Requirements for IPv6 Customer Edge Routers*, November 2013
339 <https://www.rfc-editor.org/info/rfc7084>

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

342 IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014
343 <https://www.rfc-editor.org/info/rfc7252>

344 IETF RFC 7301, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation
345 Extension*, July 2014
346 <https://www.rfc-editor.org/info/rfc7301>

347 IETF RFC 7346, *IPv6 Multicast Address Scopes*, August 2014
348 <https://www.rfc-editor.org/info/rfc7346>

349 IETF RFC 7595, *Guidelines and Registration Procedures for URI Schemes*, June 2015
350 <https://www.rfc-editor.org/info/rfc7595>

351 IETF RFC 7641, *Observing Resources in the Constrained Application Protocol
352 (CoAP)*, September 2015
353 <https://www.rfc-editor.org/info/rfc7641>

354 IETF RFC 7721, *Security and Privacy Considerations for IPv6 Address Generation Mechanisms*,
355 March 2016
356 <https://www.rfc-editor.org/info/rfc7721>

357 IETF RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*, August
358 2016
359 <https://www.rfc-editor.org/info/rfc7959>

360 IETF RFC 8075, *Guidelines for Mapping Implementations: HTTP to the Constrained Application
361 Protocol (CoAP)*, February 2017
362 <https://www.rfc-editor.org/info/rfc8075>

363 IETF RFC 8288, *Web Linking*, October 2017
364 <https://www.rfc-editor.org/info/rfc8288>

365 IETF RFC 8323, *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*,
366 February 2018
367 <https://www.rfc-editor.org/info/rfc8323>

368 IANA ifType-MIB Definitions
369 <https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib>

370 IANA IPv6 Multicast Address Space Registry
371 <http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>

372 IANA Link Relations, October 2017
373 <http://www.iana.org/assignments/link-relations/link-relations.xhtml>

374 JSON Schema Validation, *JSON Schema: interactive and non-interactive validation*, January 2013
375 <http://json-schema.org/draft-04/json-schema-validation.html>

376 OpenAPI specification, *fka Swagger RESTful API Documentation Specification*, Version 2.0
377 <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md>

378 **3 Terms, definitions, and abbreviated terms**

379 **3.1 Terms and definitions**

380 For the purposes of this document, the terms and definitions given in the following apply.

381 ISO and IEC maintain terminological databases for use in standardization at the following
382 addresses:

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

385 **3.1.1**

386 **Atomic Measurement**

387 a design pattern that ensures that the Client (3.1.6) can only access the Properties (3.1.33) of
388 linked Resources (3.1.31) atomically, that is as a single group

389 **3.1.2**

390 **Bridged Client**

391 logical entity that accesses data via a Bridged Protocol (3.1.4)

392 Note 1 to entry: For example, an AllJoyn Consumer application is a Bridged Client (3.1.2)

393 **3.1.3**

394 **Bridged Device**

395 Bridged Client (3.1.2) or Bridged Server (3.1.5)

396 **3.1.4**

397 **Bridged Protocol**

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

399 **3.1.5**

400 **Bridged Server**

401 logical entity that provides data via a Bridged Protocol (3.1.4)

402 Note 1 to entry: For example an AllJoyn Producer is a Bridged Server (3.1.5).

403 Note 2 to entry: More than one Bridged Server (3.1.5) can exist on the same physical platform.

404 **3.1.6**

405 **Client**

406 a logical entity that accesses a Resource (3.1.31) on a Server (3.1.36)

407 **3.1.7**

408 **Collection**

409 a Resource (3.1.31) that contains zero or more Links (3.1.21)

410 **3.1.8**
411 **Common Properties**
412 Properties (3.1.33) specified for all Resources (3.1.31)

413 **3.1.9**
414 **Composite Device**
415 a Device (3.1.13) that is modelled as multiple Device Types (3.1.14); with each component Device
416 Type (3.1.14) being exposed as a Collection (3.1.7)

417 **3.1.10**
418 **Configuration Source**
419 a cloud or service network or a local read-only file which contains and provides configuration
420 related information to the Devices (3.1.13)

421 **3.1.11**
422 **Core Resources**
423 those Resources (3.1.31) that are defined in this document

424 **3.1.12**
425 **Default OCF Interface**
426 an OCF Interface (3.1.18) used to generate the response when an OCF Interface (3.1.18) is omitted
427 in a request

428 **3.1.13**
429 **Device**
430 a logical entity that assumes one or more roles, e.g., Client (3.1.6), Server (3.1.36)

431 Note 1 to entry: More than one Device (3.1.13) can exist on a Platform (3.1.30).

432 **3.1.14**
433 **Device Type**
434 a uniquely named definition indicating a minimum set of Resource Types (3.1.34) that a Device
435 (3.1.13) supports

436 Note 1 to entry: A Device Type (3.1.14) provides a hint about what the Device (3.1.13) is, such as a light or a fan, for
437 use during Resource (3.1.31) discovery.

438 **3.1.15**
439 **Discoverable Resource**
440 a Resource (3.1.31) that is listed in "/oic/res"

441 **3.1.16**
442 **OCF Endpoint**
443 entity participating in the OCF protocol, further identified as the source or destination of a request
444 and response messages for a given Transport Protocol Suite

445 Note 1 to entry: Example of a Transport Protocol Suite would be CoAP over UDP over IPv6.

446 **3.1.17**
447 **Framework**
448 a set of related functionalities and interactions defined in this document, which enable
449 interoperability across a wide range of networked devices, including IoT

450 **3.1.18**
451 **OCF Interface**
452 interface description in accordance with IETF RFC 6690 and as defined by OCF that provides a
453 view to and permissible responses from a Resource (3.1.31)

454 **3.1.19**
455 **Introspection**
456 mechanism to determine the capabilities of the hosted Resources (3.1.31) of a Device (3.1.13)

457 **3.1.20**
458 **Introspection Device Data (IDD)**
459 data that describes the payloads per implemented method of the Resources (3.1.31) that make up
460 the Device (3.1.13)

461 Note 1 to entry: See 11.4 for all requirements and exceptions.

462 **3.1.21**
463 **Links**
464 extends typed web links according to IETF RFC 8288

465 **3.1.22**
466 **Non-Discoverable Resource**
467 a Resource (3.1.31) that is not listed in "/oic/res"

468 Note 1 to entry: The Resource (3.1.31) can be reached by a Link (3.1.21) which is conveyed by another Resource
469 (3.1.31). For example a Resource (3.1.31) linked in a Collection (3.1.7) does not have to be listed in "/oic/res", since
470 traversing the Collection (3.1.7) would discover the Resource (3.1.31) implemented on the Device (3.1.13).

471 **3.1.23**
472 **Notification**
473 the mechanism to make a Client (3.1.6) aware of state changes in a Resource (3.1.31)

474 **3.1.24**
475 **Observe**
476 the act of monitoring a Resource (3.1.31) by sending a RETRIEVE operation which is cached by
477 the Server (3.1.36) hosting the Resource (3.1.31) and reprocessed on every change to that
478 Resource (3.1.31)

479 **3.1.25**
480 **OpenAPI 2.0**
481 Resource (3.1.31) and Introspection Device Data (3.1.20) definitions used in this document as
482 defined in the OpenAPI specification

483 **3.1.26**
484 **Parameter**
485 an element that provides metadata about a Resource (3.1.31) referenced by the target URI of a
486 Link (3.1.21)

487 **3.1.27**
488 **Partial UPDATE**
489 an UPDATE operation to a Resource (3.1.31) that includes a subset of the Properties (3.1.33) that
490 are visible via the OCF Interface (3.1.18) being applied for the Resource Type (3.1.34)

491 **3.1.28**
492 **Permanent Immutable ID**
493 an identity for a Device (3.1.13) that cannot be altered

494 **3.1.29**
495 **Physical Device**
496 the physical thing on which a Device(s) (3.1.13) is exposed

497 **3.1.30**
498 **Platform**
499 a Physical Device (3.1.29) containing one or more Devices (3.1.13)

500 **3.1.31**
501 **Resource**
502 represents an entity modelled and exposed by the Framework (3.1.17)

503 **3.1.32**
504 **Resource Interface**
505 a qualification of the permitted requests on a Resource (3.1.31)

506 **3.1.33**
507 **Property**
508 a significant aspect or Parameter (3.1.26) of a Resource (3.1.31), including metadata, that is
509 exposed through the Resource (3.1.31)

510 **3.1.34**
511 **Resource Type**
512 a uniquely named definition of a class of Properties (3.1.33) and the interactions that are supported
513 by that class

514 Note 1 to entry: Each Resource (3.1.31) has a Property (3.1.33) "rt" whose value is the unique name of the Resource
515 Type (3.1.34).

516 **3.1.35**
517 **Secure OCF Endpoint**
518 an OCF Endpoint (3.1.16) with a secure connection (e.g., CoAPS)

519 **3.1.36**
520 **Semantic Tag**
521 meta-information that provides additional contextual information with regard to the Resource
522 (3.1.31) that is the target of a Link (3.1.21)

523 **3.1.37**
524 **Server**
525 a Device (3.1.13) with the role of providing Resource (3.1.31) state information and facilitating
526 remote interaction with its Resources (3.1.31)

527 **3.1.38**
528 **Unsecure OCF Endpoint**
529 an OCF Endpoint () with an unsecure connection (e.g., CoAP)

530 **3.1.39**
531 **Vertical Resource Type**
532 a Resource Type (3.1.34) in a vertical domain specification

533 Note 1 to entry: An example of a Vertical Resource Type (3.1.39) would be "oic.r.switch.binary".

534 **3.1.40**
535 **Virtual OCF Client**
536 logical representation of a Bridged Client (3.1.2), which an Bridged Device (3.1.3) exposes to
537 Servers (3.1.36)

538 **3.1.41**
539 **Virtual OCF Device (or VOD)**
540 Virtual OCF Client (3.1.40) or Virtual OCF Server (3.1.42)

541 **3.1.42**
542 **Virtual OCF Server**
543 logical representation of a Bridged Server (3.1.5), which an Bridged Device (3.1.3) exposes to
544 Clients (3.1.6)

545 **3.2 Abbreviated terms**
546 **3.2.1**
547 **ACL**
548 Access Control List
549 Note 1 to entry: The details are defined in ISO/IEC 30118-2:2018.
550 **3.2.2**
551 **BLE**
552 Bluetooth Low Energy
553 **3.2.3**
554 **CBOR**
555 Concise Binary Object Representation
556 **3.2.4**
557 **CoAP**
558 Constrained Application Protocol
559 **3.2.5**
560 **CoAPS**
561 Secure Constrained Application Protocol
562 **3.2.6**
563 **DTLS**
564 Datagram Transport Layer Security
565 Note 1 to entry: The details are defined in IETF RFC 6347.
566 **3.2.7**
567 **EXI**
568 Efficient XML Interchange
569 **3.2.8**
570 **IP**
571 Internet Protocol
572 **3.2.9**
573 **IRI**
574 Internationalized Resource Identifiers
575 **3.2.10**
576 **ISP**
577 Internet Service Provider
578 **3.2.11**
579 **JSON**
580 JavaScript Object Notation
581 **3.2.12**
582 **mDNS**
583 Multicast Domain Name Service
584 **3.2.13**
585 **MTU**
586 Maximum Transmission Unit

587 **3.2.14**
588 **NAT**
589 Network Address Translation

590 **3.2.15**
591 **OCF**
592 Open Connectivity Foundation

593 the organization that created this document

594 **3.2.16**
595 **REST**
596 Representational State Transfer

597 **3.2.17**
598 **RESTful**
599 REST-compliant Web services

600 **3.2.18**
601 **UDP**
602 User Datagram Protocol

603 Note 1 to entry: The details are defined in IETF RFC 768.

604 **3.2.19**
605 **URI**
606 Uniform Resource Identifier

607 **3.2.20**
608 **URN**
609 Uniform Resource Name

610 **3.2.21**
611 **UTC**
612 Coordinated Universal Time

613 **3.2.22**
614 **UUID**
615 Universal Unique Identifier

616 **3.2.23**
617 **XML**
618 Extensible Markup Language

619 **4 Document conventions and organization**

620 **4.1 Conventions**

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

625 **4.2 Notation**

626 In this document, features are described as required, recommended, allowed or DEPRECATED as
627 follows:

628 Required (or shall or mandatory)(M).

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

632 Recommended (or should)(S).

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

639 Allowed (may or allowed)(O).

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

643 DEPRECATED.

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

650 Conditionally allowed (CA).

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

653 Conditionally required (CR).

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

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

658 Words that are emphasized are printed in italic.

659 In all of the Property and Resource definition tables that are included throughout this document the
660 "Mandatory" column indicates that the item detailed is mandatory to implement; the mandating of
661 inclusion of the item in a Resource Payload associated with a CRUDN action is dependent on the
662 applicable schema for that action.

663 **4.3 Data types**

664 Resources are defined using data types derived from JSON values as defined in IETF RFC 7159.
665 However, a Resource can overload a JSON defined value to specify a particular subset of the
666 JSON value, using validation keywords defined in JSON Schema Validation.

667 Among other validation keywords, clause 7 in JSON Schema Validation defines a "format" keyword
668 with a number of format attributes such as "uri" and "date-time", and a "pattern" keyword with a
669 regular expression that can be used to validate a string. This clause defines patterns that are
670 available for use in describing OCF Resources. The pattern names can be used in document text
671 where JSON format names can occur. The actual JSON schemas shall use the JSON type and
672 pattern instead.

673 For all rows defined in Table 1, the JSON type is string.

674

Table 1 – Additional OCF Types

Pattern Name	Pattern	Description
"csv"	<none>	A comma separated list of values encoded within a string. The value type in the csv is described by the Property where the csv is used. For example a csv of integers. NOTE csv is considered deprecated and an array of strings should be used instead for new Resources.
"date"	^([0-9]{4})-(1[0-2] 0[1-9])-(3[0-1] 2[0-9] 1[0-9] 0[1-9])\$	The full-date format pattern according to IETF RFC 3339
"duration"	^(P(?:\$)([0-9]+Y)?([0-9]+M)?([0-9]+W)?([0-9]+D)?(T(?:=[0-9]+[HMS])([0-9]+H)?([0-9]+M)?([0-9]+S)?))?)\$ ^P([0-9]+W)\$ ^P([0-9]{4})-(1[0-2] 0[1-9])-(3[0-1] 2[0-9] 1[0-9] 0[1-9])T(2[0-3] 1[0-9] 0[1-9]):([0-5][0-9]):([0-5][0-9])\$ ^P([0-9]{4})(1[0-2] 0[1-9])(3[0-1] 2[0-9] 1[0-9] 0[1-9])T(2[0-3] 1[0-9] 0[1-9])([0-5][0-9])([0-5][0-9])\$	A string representing duration formatted as defined in ISO 8601. Allowable formats are: P[n]Y[n]M[n]DT[n]H[n]M[n]S, P[n]W, P[n]Y[n]-M[n]-DT[0-23]H[0-59]:M[0-59]:S, and P[n]W, P[n]Y[n]M[n]DT[0-23]H[0-59]M[0-59]S. P is mandatory, all other elements are optional, time elements must follow a T.
"int64"	^0 (-?[1-9][0-9]{0,18})\$	A string instance is valid against this attribute if it contains an integer in the range $[-(2^{63}), (2^{63})-1]$ NOTE IETF RFC 7159 clause 6 explains that JSON integers outside the range $[-(2^{53})+1, (2^{53})-1]$ are not interoperable and so JSON numbers cannot be used for 64-bit numbers.
"language-tag"	^[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*\$	An IETF language tag formatted according to IETF RFC 5646 clause 2.1.
"uint64"	^0 ([1-9][0-9]{0,19})\$	A string instance is valid against this attribute if it contains an integer in the range $[0, (2^{64})-1]$ Also see note for "int64"
"uuid"	^[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}\$	A UUID string representation formatted according to IETF RFC 4122 clause 3.

675

676 Strings shall be encoded as UTF-8 unless otherwise specified.

677 In a JSON schema, "maxLength" for a string indicates the maximum number of characters not
 678 octets. However, "maxLength" shall also indicate the maximum number of octets. If no "maxLength"
 679 is defined for a string, then the maximum length shall be 64 octets.

680 **5 Architecture**

681 **5.1 Overview**

682 The architecture enables resource based interactions among IoT artefacts, i.e. physical devices or
683 applications. The architecture leverages existing industry standards and technologies and provides
684 solutions for establishing connections (either wireless or wired) and managing the flow of
685 information among Devices, regardless of their form factors, operating systems or service providers.

686 Specifically, the architecture provides:

- 687 – A communication and interoperability framework for multiple market segments (Consumer,
688 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,
689 transports and use cases.
- 690 – A common and consistent model for describing the environment and enabling information and
691 semantic interoperability.
- 692 – Common communication protocols for discovery and connectivity.
- 693 – Common security and identification mechanisms.
- 694 – Opportunity for innovation and product differentiation.
- 695 – A scalable solution addressing different Device capabilities, applicable to smart devices as well
696 as the smallest connected things and wearable devices.

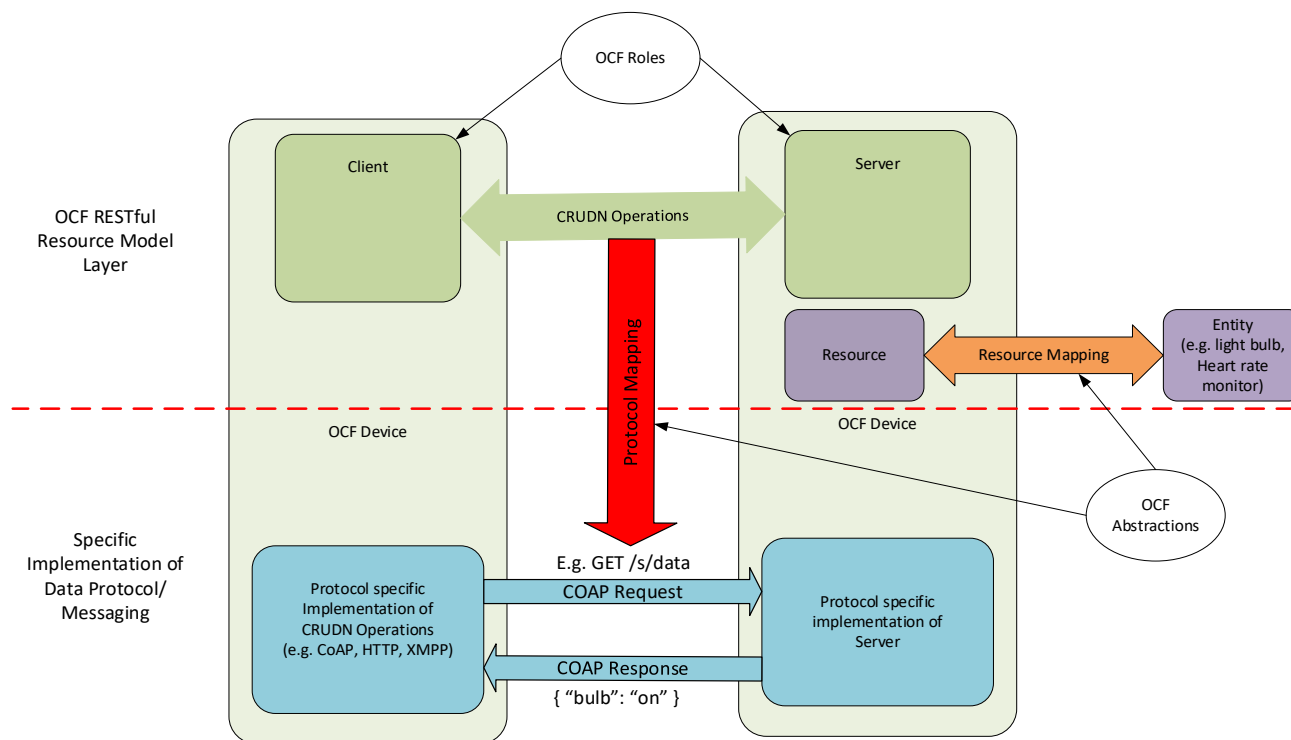
697 The architecture is based on the Resource Oriented Architecture design principles and described
698 in the 5.2 through 5.4 respectively. 5.2 presents the guiding principles for OCF operations. 5.3
699 defines the functional block diagram and Framework.

700 **5.2 Principle**

701 In the architecture, Entities in the physical world (e.g., temperature sensor, an electric light or a
702 home appliance) are represented as Resources. Interactions with an entity are achieved through
703 its Resource representations (see 7.6.3.9) using operations that adhere to Representational State
704 Transfer (REST) architectural style, i.e., RESTful interactions.

705 The architecture defines the overall structure of the Framework as an information system and the
706 interrelationships of the Entities that make up OCF. Entities are exposed as Resources, with their
707 unique identifiers (URIs) and support interfaces that enable RESTful operations on the Resources.
708 Every RESTful operation has an initiator of the operation (the Client) and a responder to the
709 operation (the Server). In the Framework, the notion of the Client and Server is realized through
710 roles. Any Device can act as a Client and initiate a RESTful operation on any Device acting as a
711 Server. Likewise, any Device that exposes Entities as Resources acts as a Server. Conformance
712 to the REST architectural style, each RESTful operation contains all the information necessary to
713 understand the context of the interaction and is driven using a small set of generic operations, i.e.,
714 CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY (CRUDN) defined in clause 8, which include
715 representations of Resources.

716 Figure 1 depicts the architecture.



717

718

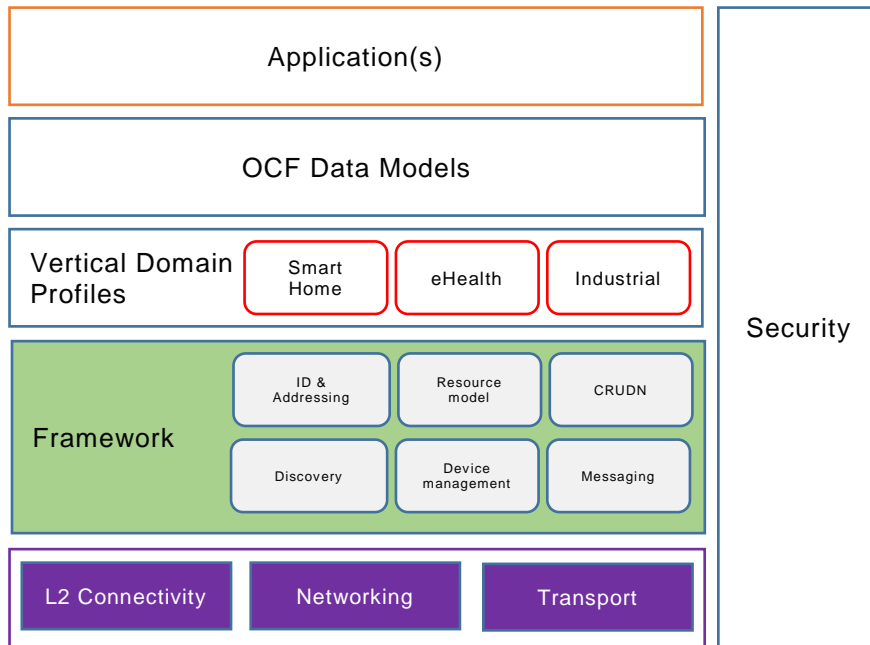
Figure 1 – Architecture - concepts

719 The architecture is organized conceptually into three major aspects that provide overall separation
720 of concern: Resource model, RESTful operations and abstractions.

- 721 – Resource model: The Resource model provides the abstractions and concepts required to
722 logically model, and logically operate on the application and its environment. The Core
723 Resource model is common and agnostic to any specific application domain such as smart
724 home, industrial or automotive. For example, the Resource model defines a Resource which
725 abstracts an entity and the representation of a Resource maps the entity's state. Other
726 Resource model concepts can be used to model other aspects, for example behaviour.
- 727 – RESTful operations: The generic CRUDN operations are defined using the RESTful paradigm
728 to model the interactions with a Resource in a protocol and technology agnostic way. The
729 specific communication or messaging protocols are part of the protocol abstraction and
730 mapping of Resources to specific protocols is provided in 11.4.
- 731 – Abstraction: The abstractions in the Resource model and the RESTful operations are mapped
732 to concrete elements using abstraction primitives. An entity handler is used to map an entity to
733 a Resource and connectivity abstraction primitives are used to map logical RESTful operations
734 to data connectivity protocols or technologies. Entity handlers may also be used to map
735 Resources to Entities that are reached over protocols that are not natively supported by OCF.

736 5.3 Functional block diagram

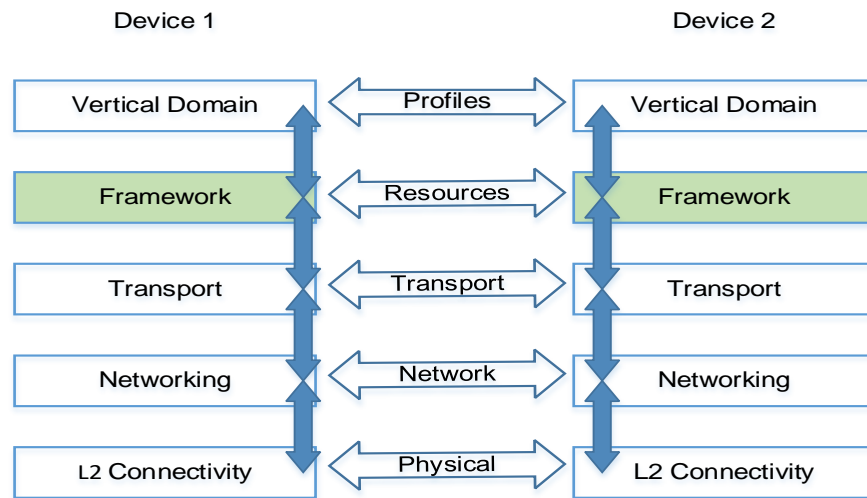
737 The functional block diagram encompasses all the functionalities required for operation. These
738 functionalities are categorized as L2 connectivity, networking, transport, Framework, and
739 application profiles. The functional blocks are depicted in Figure 2.



741

Figure 2 – Functional block diagram

- 742 – *L2 connectivity*: Provides the functionalities required for establishing physical and data link
 743 layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.
- 744 – *Networking*: Provides functionalities required for Devices to exchange data among themselves
 745 over the network (e.g., Internet).
- 746 – *Transport*: Provides end-to-end flow transport with specific QoS constraints. Examples of a
 747 transport protocol include TCP and UDP or new Transport protocols under development in the
 748 IETF, e.g., Delay Tolerant Networking (DTN).
- 749 – *Framework*: Provides the core functionalities as defined in this document. The functional block
 750 is the source of requests and responses that are the content of the communication between
 751 two Devices.
- 752 – *Vertical Domain profile*: Provides market segment specific functionalities, e.g., functions for the
 753 smart home market segment.
- 754 When two Devices communicate with each other, each functional block in a Device interacts with
 755 its counterpart in the peer Device as shown in Figure 3.



756
757 **Figure 3 – Communication layering model**

758 **5.4 Framework**

759 Framework consists of functions which provide core functionalities for operation.

- 760 – *Identification and addressing*. Defines the identifier and addressing capability. The Identification and addressing function is defined in clause 6.
- 761
- 762 – *Discovery*. Defines the process for discovering available.
- 763 – Devices (OCF Endpoint Discovery in clause 10) and
- 764 – Resources (Resource discovery in 11.2).
- 765 – *Resource model*. Specifies the capability for representation of entities in terms of Resources and defines mechanisms for manipulating the Resources. The Resource model function is defined in clause 7.
- 766
- 767
- 768 – *CRUDN*. Provides a generic scheme for the interactions between a Client and Server as defined in clause 8.
- 769
- 770 – *Messaging*. Provides specific message protocols for RESTful operation, i.e. CRUDN. For example, CoAP is a primary messaging protocol. The messaging function is defined in 11.5.
- 771
- 772 – *Security*. Includes authentication, authorization, and access control mechanisms required for secure access to Entities. The security function is defined in clause 13.
- 773

774 **6 Identification and addressing**

775 **6.1 Introduction**

776 Facilitating proper and efficient interactions between elements in the Framework, requires a means to identify, name and address these elements.

778 The *identifier* unambiguously identifies an element in a context or domain. The context or domain may be determined by the use or the application. The identifier is expected to be immutable over the lifecycle of that element and is unambiguous within a context or domain.

781 The *address* is used to define a place, way or means of reaching or accessing the element in order to interact with it. An address may be mutable based on the context.

783 The *name* is a handle that distinguishes the element from other elements in the Framework. The
784 name may be changed over the lifecycle of that element.

785 There may be methods or resolution schemes that allow determining any of these based on the
786 knowledge of one or more of others (e.g., determine name from address or address from name).

787 Each of these aspects may be defined separately for multiple contexts (e.g., a context could be a
788 layer in a stack). So an address may be a URL for addressing Resource and an IP address for
789 addressing at the connectivity layer. In some situations, both these addresses would be required.
790 For example, to do RETRIEVE (see 8.3) operation on a particular Resource representation, the
791 Client needs to know the address of the target Resource and the address of the Server through
792 which the Resource is exposed.

793 In a context or domain of use, a name or address could be used as identifier or vice versa. For
794 example, a URL could be used as an identifier for a Resource and designated as a URI.

795 The remainder of this clause discusses the identifier, address and naming from the point of view
796 of the Resource model and the interactions to be supported by the Resource model. Examples of
797 interactions are the RESTful interactions, i.e. CRUDN operation (clause 8) on a Resource. Also
798 the mapping of these to transport protocols, e.g., CoAP is described.

799 **6.2 Identification**

800 **6.2.1 Device and Platform identification**

801 This document defines three identifiers that are used for identification of the Device. All identifiers
802 are exposed via Resources that are also defined within this document (see clause 11.2).

803 The Permanent Immutable ID ("piid" Property of "/oic/d") is the immutable identity of the Device,
804 the persistent valid value of this property is typically only visible after the Device is on-boarded
805 (when not on-boarded the Device typically exposes a temporary value). This value does not change
806 across the life-cycle of the Device.

807 The Device ID ("di" Property of "/oic/d") is a mutable identity. The value changes each time the
808 Device is on-boarded. It reflects a specific on-boarded instance of the Device.

809 The Platform ID ("pi" Property of "/oic/p") is the immutable identity of the Platform on which the
810 Device is resident. When multiple logical Devices are exposed on a single Platform (for example,
811 on a Bridge) then the "pi" exposed by each Device should be the same.

812 **6.2.2 Resource identification and addressing**

813 A Resource may be identified using a URI and addressed by the same URI if the URI is a URL. In
814 some cases a Resource may need an identifier that is different from a URI; in this case, the
815 Resource may have a Property whose value is the identifier. When the URI is in the form of a URL,
816 then the URI may be used to address the Resource.

817 An OCF URI is based on the general form of a URI as defined in IETF RFC 3986 as follows:

818 `<scheme>://<authority>/<path>?<query>`

819 Specifically the OCF URI is specified in the following form:

820 `ocf://<authority>/<path>?<query>`

821 The following is a description of values that each component takes.

822 The *scheme* for the URI is "ocf". The "ocf" scheme represents the semantics, definitions and use
823 as defined in this document. If a URI has the portion preceding the "/" (double slash) omitted, then
824 the "ocf" scheme shall be assumed.

825 Each transport binding is responsible for specifying how an OCF URI is converted to a transport
826 protocol URI before sending over the network by the requestor. Similarly on the receiver side, each
827 transport binding is responsible for specifying how an OCF URI is converted from a transport
828 protocol URI before handing over to the Resource model layer on the receiver.

829 The authority of an OCF URI shall be the Device ID ("di") value, as defined in [OCF Security], of
830 the Server.

831 The *path* is a string that unambiguously identifies or references a Resource within the context of
832 the Server. In this version of the document, a path shall not include pct-encoded non-ASCII
833 characters or NUL characters. A *path* shall be preceded by a "/" (slash). The *path* may have "/"
834 (slash) separated segments for human readability reasons. In the OCF context, the "/" (slash)
835 separated segments are treated as a single string that directly references the Resources (i.e. a flat
836 structure) and not parsed as a hierarchy. On the Server, the path or some substring in the path
837 may be shortened by using hashing or some other scheme provided the resulting reference is
838 unique within the context of the host.

839 Once a path is generated, a Client accessing the Resource or recipient of the URI should use that
840 path as an opaque string and should not parse to infer a structure, organization or semantic.

841 A query string shall contain a list of "<name>=<value>" segments (aka name-value pair) each
842 separated by a "&" (ampersand). The query string will be mapped to the appropriate syntax of the
843 protocol used for messaging. (e.g., CoAP).

844 A URI may be either fully qualified or relative generation of URI.

845 A URI may be defined by the Client which is the creator of that Resource. Such a URI may be
846 relative or absolute (fully qualified). A relative URI shall be relative to the Device on which it is
847 hosted. Alternatively, a URI may be generated by the Server of that Resource automatically based
848 on a pre-defined convention or organization of the Resources, based on an OCF Interface, based
849 on some rules or with respect to different roots or bases.

850 The absolute path reference of a URI is to be treated as an opaque string and a Client should not
851 infer any explicit or implied structure in the URI – the URI is simply an address. It is also
852 recommended that Devices hosting a Resource treat the URI of each Resource as an opaque string
853 that addresses only that Resource. (e.g., URI's "/a" and "/a/b" are considered as distinct addresses
854 and Resource b cannot be construed as a child of Resource a).

855 **6.3 Namespace:**

856 The relative URI prefix "/oic/" is reserved as a namespace for URIs defined in OCF specifications
857 and shall not be used for URIs that are not defined in OCF specifications. The prefix "oic." used for
858 OCF Interfaces and Resource Types is reserved for OCF specification usage.

859 **6.4 Network addressing**

860 The following are the addresses used in this document:

861 IP address

- 862 – An IP address is used when the Device is using an IP configured interface.
- 863 – When a Device only has the identity information of its peer, a resolution mechanism is needed
864 to map the identifier to the corresponding address.

865 7 Resource model

866 7.1 Introduction

867 The Resource model defines concepts and mechanisms that provide consistency and core
868 interoperability between Devices in the OCF ecosystems. The Resource model concepts and
869 mechanisms are then mapped to the transport protocols to enable communication between the
870 Devices – each transport provides the communication protocol interoperability. The Resource
871 model, therefore, allows for interoperability to be defined independent of the transports.

872 In addition, the concepts in the Resource model support modelling of the primary artefacts and
873 their relationships to one and another and capture the semantic information required for
874 interoperability in a context. In this way, OCF goes beyond simple protocol interoperability to
875 capture the rich semantics required for true interoperability in Wearable and Internet of Things
876 ecosystems.

877 The primary concepts in the Resource model are: entity, Resources, Uniform Resource Identifiers
878 (URI), Resource Types, Properties, Representations, OCF Interfaces, Collections and Links. In
879 addition, the general mechanisms are CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY.
880 These concepts and mechanisms may be composed in various ways to define the rich semantics
881 and interoperability needed for a diverse set of use cases that the Framework is applied to.

882 In the OCF Resource model Framework, an entity needs to be visible, interacted with or
883 manipulated, it is represented by an abstraction called a Resource. A Resource encapsulates and
884 represents the state of an entity. A Resource is identified, addressed and named using URIs.

885 Properties are "key=value" pairs and represent state of the Resource. A snapshot of these
886 Properties is the Representation of the Resource. A specific view of the Representation and the
887 mechanisms applicable in that view are specified as OCF Interfaces. Interactions with a Resource
888 are done as Requests and Responses containing Representations.

889 A Resource instance is derived from a Resource Type. The uni-directional relationship between
890 one Resource and another Resource is defined as a Link. A Resource that has Properties and
891 Links is a Collection.

892 A set of Properties can be used to define a state of a Resource. This state may be retrieved or
893 updated using appropriate Representations respectively in the response from and request to that
894 Resource.

895 A Resource (and Resource Type) could represent and be used to expose a capability. Interactions
896 with that Resource can be used to exercise or use that capability. Such capabilities can be used to
897 define processes like discovery, management, advertisement etc. For example: *discovery of*
898 *Resources on a Device* can be defined as the retrieval of a representation of a specific Resource
899 where a Property or Properties have values that describe or reference the Resources on the Device.

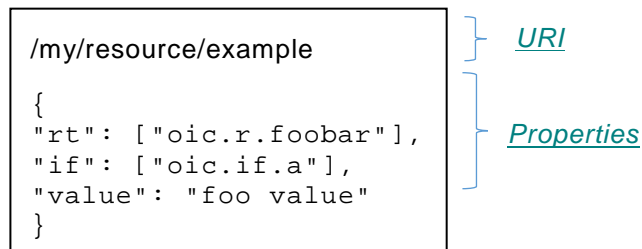
900 The information for Request or Response with the Representation may be communicated on the
901 wire by serializing using a transfer protocol or encapsulated in the payload of the transport protocol
902 – the specific method is determined by the normative mapping of the Request or Response to the
903 transport protocol. See 11.4 for transport protocols supported.

904 The OpenAPI 2.0 definitions (Annex A) used in this document are normative. This includes that all
905 defined JSON payloads shall comply with the indicated OpenAPI 2.0 definitions. Annex A contains
906 all of the OpenAPI 2.0 definitions for Resource Types defined in this document.

907 **7.2 Resource**

908 A Resource shall be defined by one or more Resource Type(s) – see Annex A for Resource Type.
909 A request to CREATE a Resource shall specify one or more Resource Types that define that
910 Resource.

911 A Resource is hosted in a Device. A Resource shall have a URI as defined in clause 6. The URI
912 may be assigned by the Authority at the creation of the Resource or may be pre-defined by the
913 definition of the Resource Type. An example Resource representation is depicted in Figure 4.



914

915 **Figure 4 – Example Resource**

916 Core Resources are the Resources defined in this document to enable functional interactions as
917 defined in clause 10 (e.g., Discovery, Device management, etc). Among the Core Resources,
918 `"/oic/res"`, `"/oic/p"`, and `"/oic/d"` shall be supported on all Devices. Devices may support other Core
919 Resources depending on the functional interactions they support.

920 **7.3 Property**

921 **7.3.1 Introduction**

922 A Property describes an aspect that is exposed through a Resource including meta-information
923 related to that Resource.

924 A Property shall have a name i.e. Property Name and a value i.e. Property Value. The Property is
925 expressed as a key-value pair where key is the Property Name and value the Property Value like
926 `<Property Name> = <Property Value>`. For example if the "temperature" Property has a Property
927 Name "temp" and a Property Value "30F", then the Property is expressed as `temp=30F`. The
928 specific format of the Property depends on the encoding scheme. For example, in JSON, Property
929 is represented as `"key": value` (e.g., `"temp": 30`).

930 In addition, the Property definition shall have a

- 931 – *Value Type* – the Value Type defines the values that a Property Value may take. The Value
932 Type may be a simple data type (e.g. string, Boolean) as defined in 4.3 or may be a complex
933 data type defined with a schema. The Value Type may define
 - 934 – Value Rules define the rules for the set of values that the Property Value may take. Such
935 rules may define the range of values, the min-max, formulas, the set of enumerated values,
936 patterns, conditional values, and even dependencies on values of other Properties. The
937 rules may be used to validate the specific values in a Property Value and flag errors.
- 938 – *Mandatory* – specifies if the Property is mandatory or not for a given Resource Type.
- 939 – *Access modes* – specifies whether the Property may be read, written or both. Updates are
940 equivalent to a write. "r" is used for read and "w" is used for write – both may be specified.
941 Write does not automatically imply read.

942 The definition of a Property may include the following additional information – these items are
943 informative:

- 944 – *Property Title* - a human-friendly name to designate the Property; usually not sent over the wire.
- 945 – *Description* – descriptive text defining the purpose and expected use of this Property.

946 In general, a Property is meaningful only within the Resource to which it is associated. However a
947 base set of Properties that may be supported by all Resources, known as Common Properties,
948 keep their semantics intact across Resources i.e. their "key=value" pair means the same in any
949 Resource. Detailed tables for all Common Properties are defined in 7.3.2.

950 **7.3.2 Common Properties**

951 **7.3.2.1 Introduction**

952 The Common Properties defined in this clause may be specified for all Resources. The following
953 Properties are defined as Common Properties:

- 954 – Resource Type
- 955 – Resource Interface
- 956 – Name
- 957 – Resource Identity.

958 The name of a Common Property shall be unique and shall not be used by other Properties. When
959 defining a new Resource Type, its non-common Properties shall not use the name of existing
960 Common Properties (e.g., "rt", "if", "n", "id"). When defining a new "Common Property", it should
961 be ensured that its name has not been used by any other Properties. The uniqueness of a new
962 Common Property name can be verified by checking all the Properties of all the existing OCF
963 defined Resource Types. However, this may become cumbersome as the number of Resource
964 Types grow. To prevent such name conflicts in the future, OCF may reserve a certain name space
965 for Common Property. Potential approaches are (1) a specific prefix (e.g. "oic") may be designated
966 and the name preceded by the prefix (e.g. "oic.psize") is only for Common Property; (2) the names
967 consisting of one or two letters are reserved for Common Property and all other Properties shall
968 have the name with the length larger than the 2 letters; (3) Common Properties may be nested
969 under specific object to distinguish themselves.

970 The ability to UPDATE a Common Property (that supports write as an access mode) is restricted
971 to the "oic.if.rw" (read-write) OCF Interface; thus a Common Property shall be updatable using the
972 read-write OCF Interface if and only if the Property supports write access as defined by the Property
973 definition and the associated schema for the read-write OCF Interface.

974 The following Common Properties for all Resources are specified in 7.3.2.2 through 7.3.2.6 and
975 summarized as follows:

- 976 – *Resource Type* ("rt") – this Property is used to declare the Resource Type of that Resource.
977 Since a Resource could be define by more than one Resource Type the Property Value of the
978 Resource Type Property can be used to declare more than one Resource type (see clause
979 7.4.4). See 7.3.2.3 for details.
- 980 – *OCF Interface* ("if") – this Property declares the OCF Interfaces supported by the Resource.
981 The Property Value of the OCF Interface Property can be multi-valued and lists all the OCF
982 Interfaces supported. See 7.3.2.4 for details.
- 983 – *Name* ("n") – the Property declares human-readable name assigned to the Resource. See
984 7.3.2.5.
- 985 – *Resource Identity* ("id"): its Property Value shall be a unique (across the scope of the host
986 Server) instance identifier for a specific instance of the Resource. The encoding of this identifier
987 is Device and implementation dependent. See 7.3.2.6 for details.

988 **7.3.2.2 Property Name and Property Value definitions**

989 The Property Name and Property Value as used in this document:

990 – *Property Name*– the key in "key=value" pair. Property Name is case sensitive and its data type
 991 is "string". Property names shall contain only letters A to Z, a to z, digits 0 to 9, hyphen, and
 992 dot, and shall not begin with a digit.

993 – *Property Value* – the value in "key=value" pair. Property Value is case sensitive when its data
 994 type is "string".

995 **7.3.2.3 Resource Type**

996 Resource Type Property is specified in 7.4.

997 **7.3.2.4 OCF Interface**

998 OCF Interface Property is specified in 7.6.

999 **7.3.2.5 Name**

1000 A human friendly name for the Resource, i.e. a specific resource instance name (e.g.,
 1001 MyLivingRoomLight), The Name Property is as defined in Table 2

1002 **Table 2 – Name Property Definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	"n"	"string"	N/A	N/A	R, W	No	Human understandable name for the Resource.

1003
 1004 The Name Property is read-write unless otherwise restricted by the Resource Type (i.e. the
 1005 Resource Type does not support UPDATE or does not support UPDATE using read-write).

1006 **7.3.2.6 Resource Identity**

1007 The Resource Identity Property shall be a unique (across the scope of the host Server) instance
 1008 identifier for a specific instance of the Resource. The encoding of this identifier is Device and
 1009 implementation dependent as long as the uniqueness constraint is met, noting that an
 1010 implementation may use a uuid as defined in 4.3. The Resource Identity Property is as defined in
 1011 Table 3.

1012 **Table 3 – Resource Identity Property Definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Identity	"id"	"string" or uuid	Implementation Dependent	N/A	R	No	Unique identifier of the Resource (over all Resources in the Device)

1013
 1014 **7.4 Resource Type**
 1015 **7.4.1 Introduction**
 1016 Resource Type is a class or category of Resources and a Resource is an instance of one or more
 1017 Resource Types.

1018 The Resource Types of a Resource is declared using the Resource Type Common Property as
 1019 described in 7.3.2.3 or in a Link using the Resource Type Parameter.

1020 A Resource Type may either be pre-defined by OCF or in custom definitions by manufacturers, end
 1021 users, or developers of Devices (vendor-defined Resource Types). Resource Types and their
 Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 20

1022 definition details may be communicated out of band (i.e. in documentation) or be defined explicitly
 1023 using a meta-language which may be downloaded and used by APIs or applications. OCF has
 1024 adopted OpenAPI 2.0 as the specification method for OCF's RESTful interfaces and Resource
 1025 definitions.

1026 Every Resource Type shall be identified with a Resource Type ID which shall be represented using
 1027 the requirements and ABNF governing the Resource Type attribute in IETF RFC 6690 (clause 2 for
 1028 ABNF and clause 3.1 for requirements) with the caveat that segments are separated by a "."
 1029 (period). The entire string represents the Resource Type ID. When defining the ID each segment
 1030 may represent any semantics that are appropriate to the Resource Type. For example, each
 1031 segment could represent a namespace. Once the ID has been defined, the ID should be used
 1032 opaquely and implementations should not infer any information from the individual segments. The
 1033 string "oic", when used as the first segment in the definition of the Resource Type ID, is reserved
 1034 for OCF-defined Resource Types. All OCF defined Resource Types are to be registered with the
 1035 IANA Core Parameters registry as described also in IETF RFC 6690.

1036 **7.4.2 Resource Type Property**

1037 A Resource when instantiated or created shall have one or more Resource Types that are the
 1038 template for that Resource. The Resource Types that the Resource conforms to shall be declared
 1039 using the "rt" Common Property for the Resource as defined in Table 4. The Property Value for the
 1040 "rt" Common Property shall be the list of Resource Type IDs for the Resource Types used as
 1041 templates (i.e., "rt"=<list of Resource Type IDs>).

1042 **Table 4 – Resource Type Common Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Type	"rt"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	Yes	The Property name rt is as described in IETF RFC 6690

1043
 1044 Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and
 1045 the host (i.e. Server) of the Resource.

1046 **7.4.3 Resource Type definition**

1047 Resource Type is specified as follows:

- 1048 – *Pre-defined URI* (optional) – a pre-defined URI may be specified for a specific Resource Type
 1049 in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that
 1050 Resource Type shall use only the pre-defined URI. An instance of a different Resource Type
 1051 shall not use the pre-defined URI.
- 1052 – *Resource Type Title* (optional) – a human friendly name to designate the Resource Type.
- 1053 – *Resource Type ID* – the value of "rt" Property which identifies the Resource Type, (e.g.,
 1054 "oic.wk.p").
- 1055 – *Resource Interfaces* – list of the OCF Interfaces that may be supported by the Resource Type.
- 1056 – *Properties* – definition of all the Properties that apply to the Resource Type. The Resource Type
 1057 definition shall define whether a property is mandatory, conditional mandatory, or optional.
- 1058 – *Related Resource Types* (optional) – the definition of other Resource Types that may be
 1059 referenced as part of the Resource Type, applicable to Collections.
- 1060 – *Mime Types* (optional) – mime types supported by the Resource including serializations (e.g.,
 1061 application/cbor, application/json, application/xml).

1062 Table 5 and Table 6 provides an example description of an illustrative foobar Resource Type and
 1063 its associated Properties.

1064 **Table 5 – Example foobar Resource Type**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction	M/CR/O
none	"foobar"	"oic.r.foobar"	"oic.if.a"	Example "foobar" Resource	Actuation	O

1065

1066 **Table 6 – Example foobar Properties**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Type	"rt"	"array"	N/A	N/A	R	Yes	Resource Type
OCF Interface	"if"	"array"	N/A	N/A	R	Yes	OCF Interface
Foo value	value	"string"	N/A	N/A	R	Yes	Foo value

1067

1068 For example, an instance of the foobar Resource Type.

```
1069 {
1070   "rt": ["oic.r.foobar"],
1071   "if": ["oic.if.a"],
1072   "value": "foo value"
1073 }
```

1074

1075 For example, a schema representation for the foobar Resource Type.

```
1076 {
1077   "$schema": "http://json-schema.org/draft-04/schema",
1078   "type": "object",
1079   "properties": {
1080     "rt": {
1081       "type": "array",
1082       "items": {
1083         "type": "string",
1084         "maxLength": 64
1085       },
1086       "minItems": 1,
1087       "readOnly": true,
1088       "description": "Resource Type of the Resource"
1089     },
1090     "if": {
1091       "type": "array",
1092       "items": {
1093         "type": "string",
1094         "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
1095 "oic.if.r", "oic.if.a", "oic.if.s"]
1096       },
1097       "value": {"type": "string"}
1098     },

```



```
1099     "required": ["rt", "if", "value"]
1100 }
```

1101 **7.4.4 Multi-value "rt" Resource**

1102 Multi-value "rt" Resource means a Resource with multiple Resource Types where none of the
1103 included Resource Types denote a well-known Resource Type (i.e. "oic.wk.<thing>"). Such a
1104 Resource is associated with multiple Resource Types and so has an "rt" Property Value of multiple
1105 Resource Type IDs (e.g. "rt": ["oic.r.switch.binary", "oic.r.light.brightness"]). The order of the
1106 Resource Type IDs in the "rt" Property Value is meaningless. For example, "rt":
1107 ["oic.r.switch.binary", "oic.r.light.brightness"] and "rt": ["oic.r.light.brightness", "oic.r.switch.binary"]
1108 have the same meaning.

1109 Resource Types for multi-value "rt" Resources shall satisfy the following conditions:

- 1110 – Property Name – Property Names for each Resource Type shall be unique (within the scope of
1111 the multi-value "rt" Resource) with the exception of Common Properties, otherwise there will be
1112 conflicting Property semantics. If two Resource Types have a Property with the same Property
1113 "Name, a multi-value "rt" Resource shall not be composed of these Resource Types.

1114 A multi-value "rt" Resource satisfies all the requirements for each Resource Type and conforms to
1115 the OpenAPI 2.0 definitions for each component Resource Type. Thus the mandatory Properties
1116 of a multi-value "rt" Resource shall be the union of all the mandatory Properties of each Resource
1117 Type. For example, mandatory Properties of a Resource with "rt": ["oic.r.switch.binary",
1118 "oic.r.light.brightness"] are "value" and "brightness", where the former is mandatory for
1119 "oic.r.switch.binary" and the latter for "oic.r.light.brightness".

1120 The multi-value "rt" Resource Interface set shall be the union of the sets of OCF Interfaces from
1121 the component Resource Types. The Resource Representation in response to a CRUDN action on
1122 an OCF Interface shall be the union of the schemas that are defined for that OCF Interface. The
1123 Default OCF Interface for a multi-value "rt" Resource shall be the baseline OCF Interface
1124 ("oic.if.baseline") as that is the only guaranteed common OCF Interface between the Resource
1125 Types.

1126 For clarity if each Resource Type supports the same set of OCF Interfaces, then the resultant multi-
1127 value "rt" Resource has that same set of OCF Interfaces with a Default OCF Interface of baseline
1128 ("oic.if.baseline").

1129 See 7.9.3 for the handling of query parameters as applied to a multi-value "rt" Resource.

1130 **7.5 Device Type**

1131 A Device Type is a class of Device. Each Device Type defined will include a list of minimum
1132 Resource Types that a Device shall implement for that Device Type. A Device may expose
1133 additional standard and vendor defined Resource Types beyond the minimum list. The Device Type
1134 is used in Resource discovery as specified in 11.2.3.

1135 Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in a
1136 Link using the Resource Type Parameter.

1137 A Device Type may either be pre-defined by an ecosystem that builds on this document, or in
1138 custom definitions by manufacturers, end users, or developers of Devices (vendor-defined Device
1139 Types). Device Types and their definition details may be communicated out of band (like in
1140 documentation).

1141 Every Device Type shall be identified with a Resource Type ID using the same syntax constraints
1142 as a Resource Type.

1143 **7.6 OCF Interface**

1144 **7.6.1 Introduction**

1145 An OCF Interface provides first a view into the Resource and then defines the requests and
1146 responses permissible on that view of the Resource. So this view provided by an OCF Interface
1147 defines the context for requests and responses on a Resource. Therefore, the same request to a
1148 Resource when targeted to different OCF Interfaces may result in different responses.

1149 An OCF Interface may be defined by either this document (a Core OCF Interface), manufacturers,
1150 end users or developers of Devices (a vendor-defined OCF Interface).

1151 The OCF Interface Property lists all the OCF Interfaces the Resource support. All Resources shall
1152 have at least one OCF Interface. The Default OCF Interface shall be defined by the Resource Type
1153 definition. The Default OCF Interface associated with all OCF-defined Resource Types shall be the
1154 supported OCF Interface listed first within the *applicable enumeration* in the definition of the
1155 Resource Type (see Annex A for the OCF-defined Resource Types defined in this document). The
1156 *applicable enumeration* is in the "parameters" enumeration referenced from the first "get" method
1157 in the first "path" in the OpenAPI 2.0 file ("post" method if no "get" exists) for the Resource Type.
1158 All Default OCF Interfaces specified in an OCF specification shall be mandatory.

1159 In addition to any defined OCF Interface in this document, all Resources shall support the baseline
1160 OCF Interface ("oic.if.baseline") as defined in 7.6.3.2.

1161 See 7.9.4 for the use of queries to enable selection of a specific OCF Interface in a request.

1162 An OCF Interface may accept more than one media type. An OCF Interface may respond with more
1163 than one media type. The accepted media types may be different from the response media types.
1164 The media types are specified with the appropriate header parameters in the transfer protocol.
1165 (NOTE: This feature has to be used judiciously and is allowed to optimize representations on the
1166 wire) Each OCF Interface shall have at least one media type.

1167

1168 **7.6.2 OCF Interface Property**

1169 The OCF Interfaces supported by a Resource shall be declared using the OCF Interface Common
1170 Property (Table 7), e.g., "if": ["oic.if.ll", "oic.if.baseline"]. The Property Value of an OCF Interface
1171 Property shall be a lower case string with segments separated by a "." (dot). The string "oic", when
1172 used as the first segment in the OCF Interface Property Value, is reserved for OCF-defined OCF
1173 Interfaces. The OCF Interface Property Value may also be a reference to an authority similar to
1174 IANA that may be used to find the definition of an OCF Interface. A Resource Type shall support
1175 one or more of the OCF Interfaces defined in 7.6.3.

1176

Table 7 – Resource Interface Property definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
OCF Interface	"if"	"array"	Array of strings, conveying OCF Interfaces	N/A	R	Yes	Property to declare the OCF Interfaces supported by a Resource.

1177

1178 **7.6.3 OCF Interface methods**

1179 **7.6.3.1 Overview**

1180 OCF Interface methods shall not violate the defined OpenAPI 2.0 definitions for the Resources as
 1181 defined in Annex A.

1182 The defined OCF Interfaces are listed in Table 8:

1183 **Table 8 – OCF standard OCF Interfaces**

OCF Interface	Name	Applicable Operations	Description
baseline	"oic.if.baseline"	RETRIEVE, NOTIFY, UPDATE ¹	The baseline OCF Interface defines a view into all Properties of a Resource including the Meta Properties. This OCF Interface is used to operate on the full Representation of a Resource.
links list	"oic.if.ll"	RETRIEVE, NOTIFY	The links list OCF Interface provides a view into Links in a Collection (Resource). Since Links represent relationships to other Resources, the links list OCF Interfaces may be used to discover Resources with respect to a context. The discovery is done by retrieving Links to these Resources. For example: the Core Resource "/oic/res" uses this OCF Interface to allow discovery of Resource hosted on a Device.
batch	"oic.if.b"	RETRIEVE, NOTIFY, UPDATE	The batch OCF Interface is used to interact with a Collection of Resources at the same time. This also removes the need for the Client to first discover the Resources it is manipulating – the Server forwards the requests and aggregates the responses
read-only	"oic.if.r"	RETRIEVE NOTIFY	The read-only OCF Interface exposes the Properties of a Resource that may be read. This OCF Interface does not provide methods to update Properties, so can only be used to read Property Values.
read-write	"oic.if.rw"	RETRIEVE, NOTIFY, UPDATE	The read-write OCF Interface exposes only those Properties that may be read from a Resource during a RETRIEVE operation and only those Properties that may be written to a Resource during and UPDATE operation.
actuator	"oic.if.a"	RETRIEVE, NOTIFY, UPDATE	The actuator OCF Interface is used to read or write the Properties of an actuator Resource.
sensor	"oic.if.s"	RETRIEVE, NOTIFY	The sensor OCF Interface is used to read the Properties of a sensor Resource.
create	"oic.if.create"	CREATE	The create OCF Interface is used to create new Resources in a Collection. Both the Resource and the Link pointing to it are created in a single atomic operation.

1184

1185 **7.6.3.2 Baseline OCF Interface**

1186 **7.6.3.2.1 Overview**

1187 The Representation that is visible using the baseline OCF Interface includes all the Properties of
 1188 the Resource including the Common Properties. The baseline OCF Interface shall be defined for
 1189 all Resource Types. All Resources shall support the baseline OCF Interface.

¹ The use of UPDATE with the baseline OCF Interface is not recommended, see clause 7.6.3.2.3.

1190 **7.6.3.2.2 Use of RETRIEVE**

1191 The baseline OCF Interface is used when a Client wants to retrieve all Properties of a Resource;
1192 that is the Server shall respond with a Resource representation that includes all of the implemented
1193 Properties of the Resource. When the Server is unable to send back the whole Resource
1194 representation, it shall reply with an error message. The Server shall not return a partial Resource
1195 representation.

1196 An example response to a RETRIEVE request using the baseline OCF Interface:

```
1197 {  
1198   "rt": ["oic.r.temperature"],  
1199   "if": ["oic.if.a", "oic.if.baseline"],  
1200   "temperature": 20,  
1201   "units": "C",  
1202   "range": [0,100]  
1203 }
```

1204 **7.6.3.2.3 Use of UPDATE**

1205 Support for the UPDATE operation using the baseline OCF Interface should not be provided by a
1206 Resource Type. Where a Resource Type needs to support the ability to be UPDATED this should
1207 only be supported using one of the other OCF Interfaces defined in Table 8 that supports the
1208 UPDATE operation.

1209 If a Resource Type is required to support UPDATE using the baseline OCF Interface, then all
1210 Properties of a Resource with the exception of Common Properties may be modified using an
1211 UPDATE operation only if the Resource Type defines support for UPDATE using baseline in the
1212 applicable OpenAPI 2.0 schema for the Resource Type. If the OCF Interfaces exposed by a
1213 Resource in addition to the baseline OCF Interface do not support the UPDATE operation, then
1214 UPDATE using the baseline OCF Interface shall not be supported.

1215 **7.6.3.3 Links list OCF Interface**

1216 **7.6.3.3.1 Overview**

1217 The Links list OCF Interface is used to provide a view into a Collection, Atomic Measurement, or
1218 "/oic.res" Resource. This view shall be an array of all Links for those Resources subject to any
1219 applied filtering being applied. The Links list OCF Interface name is "oic.if.ll".

1220 **7.6.3.3.2 Use with RETRIEVE**

1221 The RETRIEVE operation is supported with the Links list OCF Interface. A successful RETRIEVE
1222 operation shall return a status code indicating success (i.e. "Content") with a payload with the
1223 Resource representation as an array of Links. If there are no Links present in a Resource
1224 representation, then an empty array list shall be returned in response to a RETRIEVE operation
1225 request.

1226 An example of a RETRIEVE operation request using the Links list OCF Interface for a Collection is
1227 as illustrated:

```
1228 RETRIEVE /scenes/scene1?if=oic.if.ll
```

1229 The RETRIEVE operation response will be the array of Links to all Resources in the Collection as
1230 illustrated:

```
1231 Response: Content  
1232 Payload:  
1233 [  
1234   {  
1235     "href": "/the/light/1",  
1236     "rt": ["oic.r.switch.binary"],  
1237     "if": ["oic.if.a", "oic.if.baseline"],
```

```

1238     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1239   },
1240   {
1241     "href": "/the/light/2",
1242     "rt": ["oic.r.switch.binary"],
1243     "if": ["oic.if.a", "oic.if.baseline"],
1244     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1245   },
1246   {
1247     "href": "/my/fan/1",
1248     "rt": ["oic.r.switch.binary"],
1249     "if": ["oic.if.a", "oic.if.baseline"],
1250     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1251   },
1252   {
1253     "href": "/his/fan/2",
1254     "rt": ["oic.r.switch.binary"],
1255     "if": ["oic.if.a", "oic.if.baseline"],
1256     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1257   }
1258 ]
1259

```

1260 Use with NOTIFY

1261 The NOTIFY operation is supported with the Links list OCF Interface. A successful NOTIFY
 1262 operation shall return a status code indicating success (i.e. "Content") with a payload with the
 1263 Resource representation as an array of Links. If there are no Links present in a Resource
 1264 representation, then an empty array list shall be returned in response to a NOTIFY operation
 1265 request. Future events that change the Resource representation (e.g. UPDATE operation) shall
 1266 return a status code indicating success (i.e. "Content") with a payload with the newly updated
 1267 Resource representation as an array of Links.

1268 An example of a NOTIFY operation request using the Links list OCF Interface for a Collection is as
 1269 illustrated:

```
1270 NOTIFY /scenes/scene1?if=oic.if.ll
```

1271 The NOTIFY operation response will be the array of Links to all Resources in the Collection as
 1272 illustrated:

```

1273 Response: Content
1274 Payload:
1275 [
1276   {
1277     "href": "/the/light/1",
1278     "rt": ["oic.r.switch.binary"],
1279     "if": ["oic.if.a", "oic.if.baseline"],
1280     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1281   },
1282   {
1283     "href": "/the/light/2",
1284     "rt": ["oic.r.switch.binary"],
1285     "if": ["oic.if.a", "oic.if.baseline"],
1286     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1287   },
1288   {
1289     "href": "/my/fan/1",
1290     "rt": ["oic.r.switch.binary"],
1291     "if": ["oic.if.a", "oic.if.baseline"],
1292     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1293   },
1294   {

```

```
1295     "href": "/his/fan/2",
1296     "rt": ["oic.r.switch.binary"],
1297     "if": ["oic.if.a", "oic.if.baseline"],
1298     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1299   }
1300 ]
1301
```

1302 Later when the "/his/fan/2" Link is removed (e.g., UPDATE operation with the Link remove OCF
1303 Interface) the response to the NOTIFY operation request is as illustrated:

```
1304 Response: Content
1305 Payload:
1306 [
1307   {
1308     "href": "/the/light/1",
1309     "rt": ["oic.r.switch.binary"],
1310     "if": ["oic.if.a", "oic.if.baseline"],
1311     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1312   },
1313   {
1314     "href": "/the/light/2",
1315     "rt": ["oic.r.switch.binary"],
1316     "if": ["oic.if.a", "oic.if.baseline"],
1317     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1318   },
1319   {
1320     "href": "/my/fan/1",
1321     "rt": ["oic.r.switch.binary"],
1322     "if": ["oic.if.a", "oic.if.baseline"],
1323     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1324   }
1325 ]
```

1326 If the result of removing a Link results in no Links being present, then an empty array list shall be
1327 sent in a notification. An example of a response with no Links being present is as illustrated:

```
1328 Response: Content
1329 Payload:
1330 [
1331 ]
```

1332 **7.6.3.3.4 Use with CREATE, UPDATE, and DELETE**

1333 The CREATE, UPDATE and DELETE operations are not allowed by the Links list OCF Interface.
1334 Attempts to perform CREATE, UPDATE or DELETE operations using the Links list OCF Interface
1335 shall return an appropriate error status code, for example "Method Not Allowed".

1336 **7.6.3.4 Batch OCF Interface**

1337 **7.6.3.4.1 Overview**

1338 The batch OCF Interface is used to interact with a Collection of Resources using a single/same
1339 Request. The batch OCF Interface can be used to RETRIEVE or UPDATE the Properties of the
1340 linked Resources with a single request.

1341 **7.6.3.4.2 General requirements for realizations of the batch OCF Interface**

1342 All realization of the batch OCF Interface adhere to the following:

- 1343 – The batch OCF Interface name is "oic.if.b"
- 1344 – A Collection Resource has linked Resources that are represented as URIs. In the "href"
1345 Property of the batch payload the URI shall be fully qualified for remote Resources and a
1346 relative reference for local Resources.

- 1347 – The original request is modified to create new requests targeting each of the linked Resources
1348 in the Collection by substituting the URI in the original request with the URI of the linked
1349 Resource. The payload in the original request is replicated in the payload of the new requests.
- 1350 – The requests shall be forwarded assuming use of the Default OCF Interface of the linked
1351 Resources.
- 1352 – Requests shall only be forwarded to linked Resources that are identified by relation types "item"
1353 or "hosts" ("hosts" is the default relation type value should the "rel" Link Parameter not be
1354 present). Requests shall not be forwarded to linked Resources that do not contain the "item" or
1355 "hosts" relation type values.
- 1356 – Properties of the Collection Resource itself may be included in payloads using "oic.if.b" OCF
1357 Interface by exposing a single Link with the link relation "self" along with "item" within the
1358 Collection, and ensuring that Link resolution cannot become an infinite loop due to recursive
1359 references. For example, if the Default OCF Interface of the Collection is "oic.if.b", then the
1360 Server might recursively include its batch representation within its batch representation, in an
1361 endless loop. See 7.6.3.4.5 for an example of use of a Link containing "rel": ["self", "item"] to
1362 include Properties of the Collection Resource, along with linked Resources, in "oic.if.b"
1363 payloads.
- 1364 – If the Default OCF Interface of a Collection Resource is exposed using the Link relation "self",
1365 and the Default OCF Interface contains Properties that expose any Links, those Properties shall
1366 not be included in a batch representation which includes the "self" Link.
- 1367 – Any request forwarded to a linked Resource that is a Collection (including a "self" Link reference)
1368 shall have the Default OCF Interface of the linked Collection Resource applied.
- 1369 – All the responses from the linked Resources shall be aggregated into a single Response to the
1370 Client. The Server may timeout the response to a time window, the Server may choose any
1371 appropriate window based on conditions.
- 1372 – If a linked Resource cannot process the request, an empty response, i.e. a JSON object with
1373 no content ("{}") as the representation for the "rep" Property, or error response should the linked
1374 Resource Type provide an error schema or diagnostic payload, shall be returned by the linked
1375 Resource. These empty or error responses for all linked Resources that exhibit an error shall
1376 be included in the aggregated response to the original Client request. See the example in
1377 7.6.3.4.5.
- 1378 – If any of the linked Resources returns an error response, the aggregated response sent to the
1379 Client shall also indicate an error (e.g. 4.xx in CoAP). If all of the linked Resources return
1380 successful responses, the aggregated response shall include the success response code.
- 1381 – The aggregated response shall be an array of objects representing the responses from each
1382 linked Resource. Each object in the response shall include at least two items: (1) the URI of
1383 the linked Resource (fully qualified for remote Resources, or a relative reference for local
1384 Resources) as "href": <URI> and (2) the individual response object or array of objects if the
1385 linked Resource is itself a Collection using "rep" as the key, e.g. "rep": { <representation of
1386 individual response> }.
- 1387 – The Client may choose to restrict the linked Resources to which the request is forwarded by
1388 including additional query parameters in the request. The Server should process any additional
1389 query parameters in a request that includes "oic.if.b" as selectors for linked Resources that are
1390 to be processed by the request.

1391 **7.6.3.4.3 Observability of the batch OCF Interface**

1392 When a Collection supports the ability to be observed using the batch OCF Interface the following
1393 apply:

- 1394 – If the Collection Resource is marked as Observable, linked Resources referenced in the
1395 Collection may be Observed using the batch OCF Interface. If the Collection Resource is not
1396 marked as Observable then the Collection cannot be Observed and Observe requests to the

1397 Collection shall be handled as defined for the case where request validation fails in clause
1398 11.3.2.4. The Observe mechanism shall work as defined in 11.3.2 with the Observe request
1399 forwarded to each of the linked Resources. All responses to the request shall be aggregated
1400 into a single response to the Client using the same representations and status codes as for
1401 RETRIEVE operations using the batch OCF Interface.

1402 – Should any one of the Observable linked Resources fail to honour the Observe request the
1403 response to the batch Observe request shall also indicate that the entire request was not
1404 honoured using the mechanism described in 11.3.2.4.

1405 – If any of the Observable Resources in a request to a Collection using the batch OCF Interface
1406 replies with an error or Observe Cancel, the Observations of all other linked Resources shall
1407 be cancelled and the error or Observe Cancel status shall be returned to the Observing Client.

1408 NOTE Behavior may be different for Links that do network requests vs. local Resources.

1409 – All notifications to the Client that initiated an Observe request using the batch OCF Interface
1410 shall use the batch representation for the Collection. This is the aggregation of any individual
1411 Observe notifications received by the Device hosting the Collection from the individual Observe
1412 requests that were forwarded to the linked Resources.

1413 – Linked Resources which are not marked Observable in the Links of a Collection shall not trigger
1414 Notifications, but may be included in the response to, and subsequent Notifications resulting
1415 from, an Observe request to the batch OCF Interface of a Collection.

1416 – Each notification shall contain the most current values for all of the Linked Resources that would
1417 be included if the original Observe request were processed again. The Server hosting the
1418 Collection may choose to RETRIEVE all of the linked Resources each time, or may choose to
1419 employ caching to avoid retrieving linked Resources on each Notification.

1420 – If a Linked Resource is Observable and has responded with a successful Observe response,
1421 the most recently reported value of that Resource is considered to be the most current value
1422 and may be reported in all subsequent Notifications.

1423 – Links in the Collection should be Observed by using the "oic.if.ll" OCF Interface. A notification
1424 shall be sent any time the contents of the "oic.if.ll" OCF Interface representation are changed;
1425 that is, if a Link is added, if a Link is removed, or if a Link is updated. Notifications on the
1426 "oic.if.ll" OCF Interface shall contain all of the Links in the "oic.if.ll" OCF Interface representation.

1427 – Other Properties of the Collection Resource, if present, may be Observed by using the OCF
1428 Interfaces defined in the definition for the Resource Type, including using the "oic.if.baseline"
1429 OCF Interface.

1430 **7.6.3.4.4 UPDATE using the batch OCF Interface**

1431 When a Collection supports the ability for the linked Resources to be the subject of the UPDATE
1432 operation using the batch OCF Interface the following apply:

1433 – A Client shall perform UPDATE operations using the batch OCF Interface by creating a payload
1434 that is similar to a RETRIEVE response payload from a batch OCF Interface request. The Server
1435 shall send a separate UPDATE request to each of the linked Resources according to each "href"
1436 Property and the corresponding value of the "rep" Property.

1437 – Items shall always contain a link-specific "href".

1438 – An UPDATE received by a Server with an empty "href" shall be rejected with a response
1439 indicating an appropriate error (e.g. bad request).

1440 – Each linked Resource shall follow the requirements for an UPDATE request may not be
1441 supported by the linked Resource. In such cases, writable Properties in the UPDATE operation
1442 as defined in clause 8.4.

1443 – The UPDATE response shall contain the updated values using the same payload schema as
1444 RETRIEVE operations if provided by the linked Resource, along with the appropriate status
1445 code. The aggregated response payload shall reflect the known state of the updated Properties

1446 after the batch update was completed. If no payload is provided by the updated Resource, then
1447 an empty response (i.e. "rep": {}) shall be provided for that Resource.

1448 – A Collection shall not support the use of the UPDATE operation to add, modify, or remove Links
1449 in an existing Collection using the "oic.if.baseline", "oic.if.rw" or "oic.if.a" OCF Interfaces.

1450 – A Collection shall not support the use of the UPDATE operation using the batch OCF Interface
1451 when the Collection contains Links that resolve to Resources that are not hosted on the Device
1452 that also hosts the Collection. If such a Collection receives an UPDATE operation, the operation
1453 shall be rejected with a response indicating an appropriate error (e.g. method not allowed). If
1454 the ability to UPDATE linked remote Resources is desired, the use of the optional scene feature
1455 (see clause 11.6 in [1]) to effect the UPDATE could be utilized.

1456 **7.6.3.4.5 Examples: Batch OCF Interface**

1457 Note that the examples provided in Table 9 are illustrative and do not include all mandatory schema
1458 elements in all cases. It is assumed that the Default OCF Interface for the Resource Type
1459 "x.org.example.rt.room" is specified in its Resource Type definition file as "oic.if.rw", which exposes
1460 the Properties "x.org.example.colour" and "x.org.example.size".

Table 9 – Batch OCF Interface Example

Resources	<pre> /a/room/1 { "rt": "x.org.example.rt.room", "if": ["oic.if.rw", "oic.if.baseline", "oic.if.b", "oic.if.ll"], "x.org.example.colour": "blue", "x.org.example.dimension": "15bx15wx10h", "links": [{ "href": "/a/room/1", "rel": ["self", "item"], "rt": ["x.org.example.rt.room"], "if": ["oic.if.rw", "oic.if.baseline", "oic.if.b", "oic.if.ll"], "p": {"bm": 2} }, { "href": "/the/light/1", "rel": ["item"], "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "ins": "11111", "p": {"bm": 2} }, { "href": "/the/light/2", "rel": ["item"], "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "ins": "22222", "p": {"bm": 2} }, { "href": "/my/fan/1", "rel": ["item"], "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "ins": "33333", "p": {"bm": 2} }, { "href": "/his/fan/2", "rel": ["item"], "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "ins": "44444", "p": {"bm": 2} }, { "href": "/the/switches/1", "rel": ["item"], "rt": ["oic.wk.col"], "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"], "ins": "55555", "p": {"bm": 2} }] } /the/light/1 { "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "value": false } /the/light/2 { "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "value": true } /my/fan/1 { "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "value": true } /his/fan/2 { "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "value": false } /the/switches/1 { "rt": ["oic.wk.col"], "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"], "links": [{ "href": "/switch-1a", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm": 2} }] } </pre>
------------------	---

	<pre>{ "href": "/switch-lb", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a","oic.if.baseline"], "p": {"bm": 2 } }]</pre>
--	---

<p>Use of batch, successful response</p>	<pre> Request: GET /a/room/1?if=oic.if.b Becomes the following individual request messages issued by the Device in the Client role GET /a/room/1 (NOTE: uses the Default OCF Interface as specified for the Collection Resource, in this example oic.if.rw) GET /the/light/1 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /the/light/2 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /my/fan/1 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /his/fan/2 (NOTE: Uses the Default OCF Interface as specified for this Resource) GET /the/switches/1 (NOTE: Uses the Default OCF Interface for the Collection that is within the Collection) Response: [{ "href": "/a/room/1", "rep": {"x.org.example.colour": "blue", "x.org.example.dimension": "15bx15wx10h"} }, { "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": true} }, { "href": "/my/fan/1", "rep": {"value": true} }, { "href": "/his/fan/2", "rep": {"value": false} }, { "href": "/the/switches/1", "rep": [{ "href": "/switch-1a", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm": 2}, "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:5555"}] }, { "href": "/switch-1b", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm": 2}, "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:5555"}] }] }] </pre>
---	---

<p>Use of batch, error response</p>	<p>Should any of the RETRIEVE requests in the previous example fail then the response includes an empty payload for that Resource instance and an error code is sent. The following example assumes errors from "/my/fan/1" and "/the/switches/1"</p> <p>Error Response:</p> <pre>[{ "href": "/a/room/1", "rep": {"x.org.example.colour": "blue", "x.org.example.dimension": "15bx15wx10h"} }, { "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": true} }, { "href": "/my/fan/1", "rep": {} }, { "href": "/his/fan/2", "rep": {"value": false} }, { "href": "/the/switches/1", "rep": {} }]</pre>
--	--

<p>Use of batch</p> <p>(UPDATE has POST semantics)</p>	<pre>UPDATE /a/room/1?if=oic.if.b [{ "href": "", "rep": { "value": false } }]</pre> <p>Since the "href" value in the UPDATE request is empty, the request is forwarded to all Resources in the Collection and becomes:</p> <pre>UPDATE /a/room/1 { "value": false } UPDATE /the/light/1 { "value": false } UPDATE /the/light/2 { "value": false } UPDATE /my/fan/1 { "value": false } UPDATE /his/fan/2 { "value": false } UPDATE /the/switches/1 { "value": false }</pre> <p>Response:</p> <pre>[{ "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": false} }, { "href": "/my/fan/1", "rep": {"value": false} }, { "href": "/his/fan/2", "rep": {"value": false} }, { "href": "/the/switches/1", "rep": { } }]</pre> <p>Since /a/room/1 does not have a "value" Property exposed by its Default OCF Interface, the UPDATE request will be silently ignored and it will not be included in the UPDATE response.</p> <p>Since the UPDATE request with the links list OCF Interface is not allowed, an empty payload for the "/the/switches/1" is included in the UPDATE response and an error code is sent.</p>
--	--

<p>Use of batch (UPDATE has POST semantics)</p>	<pre> UPDATE /a/room/1?if=oic.if.b [{ "href": "/the/light/1", "rep": { "value": false } }, { "href": "/the/light/2", "rep": { "value": true } }, { "href": "/a/room/1", "rep": { "x.org.example.colour": "red" } }] </pre> <p>This turns /the/light/1 off, turns /the/light/2 on, and sets the colour of /a/room/1 to "red".</p> <p>The response will be same as response for GET /a/room/1?if=oic.if.b with the updated Property values as shown.</p> <pre> [{ "href": "/a/room/1", "rep": {"x.org.example.colour": "red", "x.org.example.dimension": "15bx15wx10h"} }, { "href": "/the/light/1", "rep": {"value": false} }, { "href": "/the/light/2", "rep": {"value": true} }] </pre> <p>Example use of additional query parameters to select items by matching Link Parameters.</p> <p>Turn on light 1 based on the "ins" Link Parameters value of "11111"</p> <pre> UPDATE /a/room/1?if=oic.if.b&ins=11111 [{ "href": "", "rep": { "value": false } }] </pre> <p>Similar to the earlier example, "href": "" applies the UPDATE request to all of the Resources in the Collection. Since the additional query parameter ins=11111 selects only links that have a matching "ins" value, only one link is selected. The payload is applied to the target Resource of that link, /the/light/1.</p>
--	--

	<p>Retrieving the item using the same query parameter: RETRIEVE /a/room/1?if=oic.if.b&ins=11111</p> <p>Response payload:</p> <pre>[{ "href": "/the/light/1", "rep": { "value": false } }]</pre>
--	--

1462

1463 **7.6.3.5 Actuator OCF Interface**

1464 The actuator OCF Interface is the OCF Interface for viewing Resources that may be actuated i.e.
1465 changes some value within or the state of the entity abstracted by the Resource:

- 1466 – The actuator OCF Interface name shall be "oic.if.a"
- 1467 – The actuator OCF Interface shall expose in the Resource Representation all mandatory
1468 Properties as defined by the applicable OpenAPI 2.0 schema; the actuator OCF Interface may
1469 also expose in the Resource Representation optional Properties as defined by the applicable
1470 OpenAPI 2.0 schema that are implemented by the target Device.

1471 For example, a "Heater" Resource (for illustration only):

```
1472 /a/act/heater
1473 {
1474   "rt": ["acme.gas"],
1475   "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
1476   "settemp": 10,
1477   "currenttemp" : 7
1478 }
```

1479 The actuator OCF Interface with respect to "Heater" Resource (for illustration only):

1480

1481 a) Retrieving values of an actuator.

1482 Request: GET /a/act/heater?if="oic.if.a"

1483

1484 Response:

```
1485 {
1486   "settemp": 10,
1487   "currenttemp" : 7
1488 }
```

1489 b) Correct use of actuator OCF Interface.

1490

1491 Request: POST /a/act/heater?if="oic.if.a"

1492

```
1493 {
1494   "settemp": 20
1495 }
```

1495 Response:

```
1496 {
1497   Ok
1498 }
```

1499 c) Incorrect use of actuator OCF Interfance.

1500

1501 Request: POST /a/act/heater?if="oic.if.a"
 1502 {
 1503 "if": ["oic.if.s"] ← this is visible through baseline OCF Interface
 1504 }
 1505 Response:
 1506 {
 1507 Error
 1508 }

1509 – A RETRIEVE request using this OCF Interface shall return the Representation for this Resource
 1510 subject to any query and filter parameters that may also exist.

1511 – An UPDATE request using this OCF Interface shall provide a payload or body that contains the
 1512 Properties that will be updated on the target Resource.

1513 7.6.3.6 Sensor OCF Interface

1514 The sensor OCF Interface is the OCF Interface for retrieving measured, sensed or capability
 1515 specific information from a Resource that senses:

- 1516 – The sensor OCF Interface name shall be "oic.if.s".
- 1517 – The sensor OCF Interface shall expose in the Resource Representation all mandatory
 1518 Properties as defined by the applicable OpenAPI 2.0 schema; the sensor OCF Interface may
 1519 also expose in the Resource Representation optional Properties as defined by the applicable
 1520 OpenAPI 2.0 schema that are implemented by the target Device.
- 1521 – A RETRIEVE request using this OCF Interface shall return this representation for the Resource
 1522 subject to any query and filter parameters that may also exist.

1523 NOTE: The example here is with respect to retrieving values of a sensor

1524 Request: GET /a/act/heater?if="oic.if.s"
 1525
 1526 Response:
 1527 {
 1528 "currenttemp": 7
 1529 }
 1530 }
 1531

1532 Incorrect use of the sensor.

1533 Request: PUT /a/act/heater?if="oic.if.s" ← PUT is not allowed
 1534 {
 1535 "settemp": 20 ← this is possible through actuator OCF Interface
 1536 }
 1537 Response:
 1538 {
 1539 Error
 1540 }
 1541

1542 Another incorrect use of the sensor.

1543 Request: POST /a/act/heater?if="oic.if.s" ← POST is not allowed
 1544 {
 1545 "currenttemp": 15 ← this is possible through actuator OCF Interface
 1546 }
 1547 Response:
 1548 {
 1549 Error
 1550 }

1551 **7.6.3.7 Read-only OCF Interface**

1552 The read-only OCF Interface exposes only the Properties that may be read. This includes
1553 Properties that may be read-only, read-write but not Properties that are write-only or set-only. The
1554 applicable operations that can be applied to a Resource are only RETRIEVE and NOTIFY. An
1555 attempt by a Client to apply a method other than RETRIEVE or NOTIFY to a Resource shall be
1556 rejected with an error response code.

1557 **7.6.3.8 Read-write OCF Interface**

1558 The read-write OCF Interface is a generic OCF Interface to support reading and setting Properties
1559 in a Resource. The applicable methods that can be applied to a Resource are only RETRIEVE,
1560 NOTIFY, and UPDATE. For the RETRIEVE and NOTIFY operations, the behaviour is the same as
1561 for the "oic.if.r" OCF Interface defined in 7.6.3.7. For the UPDATE operation, read-only Properties
1562 (i.e. Properties tagged with "readOnly=True" in the OpenAPI 2.0 definition) shall not be in the
1563 UPDATE payload. An attempt by a Client to apply a method other than RETRIEVE, NOTIFY, or
1564 UPDATE to a Resource shall be rejected with an error response code.

1565 **7.6.3.9 Create OCF Interface**

1566 **7.6.3.9.1 Overview**

1567 The create OCF Interface is used to create Resource instances in a Collection. An instance of a
1568 Resource and the Link pointing to the Resource are created together, atomically, according to a
1569 Client-supplied representation. The create OCF Interface name is "oic.if.create". A Collection which
1570 exposes the "oic.if.create" OCF Interface shall expose the "rts" Property (see clause 7.8.2.8) with
1571 all Resource Types that can be hosted with the Collection. If a Client attempts to create a Resource
1572 Type which is not supported by the Collection, the Server shall return an appropriate error status
1573 code, for example "Bad Request". Successful CREATE operations shall return a success code, i.e.
1574 "Created". The IDD for all allowed Resource Types that may be created shall adhere to
1575 Introspection for dynamic Resources (see clause 11.4).

1576 **7.6.3.9.2 Data format for CREATE**

1577 The data format for the create OCF Interface is similar to the data format for the batch OCF
1578 Interface. The create OCF Interface format consists of a set of Link Parameters and a "rep"
1579 Parameter which contains a representation for the created Resource.

1580 The representation supplied for the Link pointing to the newly created Resource shall contain at
1581 least the "rt" and "if" Link Parameters.

1582 The Link Parameter "p" should be included in representations supplied for all created Resources.
1583 If the "Discoverable" bit is set, then the supplied Link representation shall be exposed in "/oic/res"
1584 of the Device on which the Resource is being created. The Link Parameters representation in the
1585 "/oic/res" Resource does not have to mirror the Link Parameters in the Collection of the created
1586 Resource (e.g., "ins" Parameter).

1587 Creating a discoverable Resource is the only way to add a Link to "/oic/res".

1588 If the "p" Parameter is not included, the Server shall create the Resource using the default settings
1589 of not discoverable, and not observable.

1590 The representation supplied for a created Resource in the value of the "rep" Parameter shall
1591 contain all mandatory Properties required by the Resource Type to be created excluding the
1592 Common Properties "rt" and "if" as they are already included in the create payload.

1593 Note that the "rt" and "if" Property Values are created from the supplied Link Parameters of the
1594 Resource creation payload.

1595 If the supplied representation does not contain all of the required Properties and Link Parameters,
1596 the Server shall return an appropriate error status code, for example "Bad Request".

1597 An example of the create OCF Interface payload is as illustrated:

```
1598 {  
1599   "rt": ["oic.r.temperature"],  
1600   "if": ["oic.if.a","oic.if.baseline"],  
1601   "p": {"bm":3},  
1602   "rep": {  
1603     "temperature": 20  
1604   }  
1605 }
```

1606 The representation returned when a Resource is successfully created shall contain the "href", "if",
1607 and "rt" Link Parameters and all other Link Parameters that were included in the CREATE operation.
1608 In addition, the "rep" Link Parameter shall include all Resource Properties as well as the "rt" and
1609 "if" Link Parameters supplied in the CREATE operation. The Server may include additional Link
1610 Parameters and Properties in the created Resource as required by the application-specific
1611 Resource Type. The Server shall assign an "ins" value to each created Link and shall include the
1612 "ins" Parameter in the representation of each created Link as illustrated in the Collection that the
1613 Link of the created Resource was created within:

```
1614 {  
1615   "href": "/3755f3ac",  
1616   "rt": ["oic.r.temperature"],  
1617   "if": ["oic.if.a","oic.if.baseline"],  
1618   "ins": 39724818,  
1619   "p": {"bm":3},  
1620   "rep": {  
1621     "rt": ["oic.r.temperature"],  
1622     "if": ["oic.if.a","oic.if.baseline"],  
1623     "temperature": 20  
1624   }  
1625 }
```

1626 The Link Parameters representation in the "/oic/res" Resource, if the created Resource is
1627 discoverable, may not mirror exactly all the Link Parameters added in the Collection; except it shall
1628 expose at a minimum the mandatory Properties of the Link (i.e., "rt", "if", and "href") of the created
1629 Resource.

1630 7.6.3.9.3 Use with CREATE

1631 The CREATE operation shall be sent to the URI of the Collection in which the Resource is to be
1632 created. The query string "?if=oic.if.create" shall be included in all CREATE operations.

1633 The Server shall generate a URI for the created Resource and include the URI in the "href"
1634 Parameter of the created Link.

1635 When a Server successfully completes a CREATE operation using the "oic.if.create" OCF Interface
1636 addressing a Collection, the Server shall automatically modify the ACL Resource to provide initial
1637 authorizations for accessing for the newly created Resource according to ISO/IEC 30118-2:2018.

1638 An example performing a CREATE operation is as illustrated:

```
1639 CREATE /scenes/scenel?if=oic.if.create  
1640 {  
1641   "rt": ["oic.r.temperature"],  
1642   "if": ["oic.if.a","oic.if.baseline"],  
1643   "p": {"bm":3},  
1644   "rep": {  
1645     "temperature": 20
```

```

1646     }
1647   }
1648   Response: Created
1649   Payload:
1650   {
1651     "href": "/3755f3ac",
1652     "ins": 39724818,
1653     "rt": ["oic.r.temperature"],
1654     "if": ["oic.if.a","oic.if.baseline"],
1655     "p": {"bm":3},
1656     "rep": {
1657       "rt": ["oic.r.temperature"],
1658       "if": ["oic.if.a","oic.if.baseline"],
1659       "temperature": 20
1660     }
1661   }

```

1662 **7.6.3.9.4 Use with UPDATE and DELETE**

1663 The UPDATE and DELETE operations are not allowed by the create OCF Interface. Attempts to
 1664 perform UPDATE or DELETE operations using the create OCF Interface shall return an appropriate
 1665 error status code, for example "Method Not Allowed", unless the UPDATE and CREATE operations
 1666 map to the same transport binding method (e.g., CoAP with the POST method). In that situation
 1667 where the UPDATE and CREATE operations map to the same transport binding method, this shall
 1668 be processed as a CREATE operation according to clause 7.6.3.9.3.

1669 **7.7 Resource representation**

1670 Resource representation captures the state of a Resource at a particular time. The Resource
 1671 representation is exchanged in the request and response interactions with a Resource. A Resource
 1672 representation may be used to retrieve or update the state of a Resource.

1673 The Resource representation shall not be manipulated by the data connectivity protocols and
 1674 technologies (e.g., CoAP, UDP/IP or BLE).

1675 **7.8 Structure**

1676 **7.8.1 Introduction**

1677 In many scenarios and contexts, the Resources may have either an implicit or explicit structure
 1678 between them. This may be achieved through the use of Collection (7.8.3) and Atomic
 1679 Measurement (7.8.4) Resources.

1680 **7.8.2 Resource relationships (Links)**

1681 **7.8.2.1 Introduction**

1682 Resource relationships are expressed as Links. A Link is a hyperlink, which defines a typed
 1683 connection between two Resources. Hyperlinks, or web links, have the following components as
 1684 defined in IETF RFC 8288:

- 1685 – Link context (URI reference) as defined in 7.8.2.2
- 1686 – Link relation type as defined in 7.8.2.3
- 1687 – Link target (URI reference) as defined in 7.8.2.4
- 1688 – Link target attributes as defined in 7.8.2.5

1689 The Link context is the Resource with which the Link is associated. A Link is viewed as a statement
 1690 of the form "(Link context) has a (Link relation type) to a Resource at (Link target), which has (Link
 1691 target attributes)" as per IETF RFC 8288 clause 2.

1692 To paraphrase, the Link target is related to the Link context according to the Link relation type.
1693 Additionally, the Link target attributes make semantic statements about the Link target, to identify
1694 the content type, physical location, etc.

1695 Links conform to the definitions in IETF RFC 8288, with an example JSON serialization with
1696 associated Link Parameters as illustrated:

```
1697 {  
1698   "anchor": "/some/ocf/resource",      // Link context, optional  
1699   "rel": ["hosts"],                  // Link relation Type, optional  
1700   "href": "/some/other/ocf/resource", // Link target, required  
1701   "p": {"bm": 3},                   // Link target attributes, optional  
1702   "if": ["oic.if.baseline"],         // Link target attributes, required  
1703   "rt": ["oic.r.sensor"]             // Link target attributes, required  
1704 }
```

1705

1706 Additional items in the Link may be made mandatory based on the use of the Links in different
1707 contexts (e.g. in Collections, in discovery, in bridging etc.). The OpenAPI 2.0 file for the Link
1708 payload is detailed in Annex A.

1709 Another example of a Link is as illustrated:

```
1710 {"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a",  
1711 "oic.if.baseline"], "p": {"bm": 3}, "rel": "item"}
```

1712 **7.8.2.2 Link context**

1713 The Link context is defined in the Link using the "anchor" Parameter. If the Link doesn't contain an
1714 "anchor" Parameter, the Link context shall be the Resource from which the Link was retrieved.

1715 **7.8.2.3 Link relation type**

1716 The Link relation type conveys the semantics of the Link. The Link relation type is defined in the
1717 Link using the "rel" Parameter. If the Link doesn't contain a "rel" Parameter, the Link relation type
1718 shall be assumed to have the default value "hosts", which means that the Resource at the Link
1719 target is "hosted" by the Resource at the Link context. The set of Link relation types to be used to
1720 describe various relationships between Resources are as listed:

- 1721 – "hosts"
 - 1722 – The Link target points to a Resource that is hosted at the Link context. This Link relation
1723 type indicates that the Resource is allowed to be included in the batch representations of
1724 the Link target. This Link relation type is defined by IETF RFC 6690.
- 1725 – "self"
 - 1726 – The Link refers to the Link context, which allows a Link to describe the Resource at the Link
1727 context, which is to say that the Link can describe the Collection or Atomic Measurement
1728 Resource that the Link is retrieved from. The Link target points to the Link context, and the
1729 Link target attributes describe the Link context. This Link relation type is defined by
1730 IETF RFC 4287.
- 1731 – "item"
 - 1732 – The Link target points to a Resource that is a member of the Collection or Atomic
1733 Measurement at the Link context, which might not specifically be hosted by the Collection
1734 or Atomic Measurement Resource, and is allowed to be contained in batch representations
1735 of the Collection or Atomic Measurement. An example is using "rel": "item" to declare that
1736 the Properties of the Collection or Atomic Measurement Resource itself should be included
1737 in a batch representation of the Collection or Atomic Measurement. This Link relation type
1738 is defined by IETF RFC 6573.

1739 All of these Link relation types are registered in the IANA Registry for Link relations types defined
 1740 in IANA Link Relations. Other Link relation types may be included in Links, provided that they
 1741 conform to the requirements in IETF RFC 8288. Other Link relation types may be defined for
 1742 features contained in other specifications and may not be included in what is defined in this clause.
 1743 The presence of Link relation types not defined in this document does not affect the processing of
 1744 Link relation types defined in this document.

1745 When there is more than one Link relation type value in a Link, all of the values apply to describe
 1746 the relationship between the Link context and the Link target. A Link with multiple Link relation type
 1747 values is equivalent to a set of Links having the same Link context and Link target, each having
 1748 one of the Link relation values.

1749 **7.8.2.4 Link target**

1750 The Link target is a URI reference to a Resource using the "href" Parameter.

1751 **7.8.2.5 Parameters for Link target attributes**

1752 **7.8.2.5.1 Introduction**

1753 Link target attributes are specialisations of Link Parameters. Table 10 lists all the Link target
 1754 attributes defined in this document.

1755 **Table 10 – Link target attributes list**

Parameter title	Parameter name	Mandatory	Description
Device ID	"di"	No	Defined in clause 7.8.2.5.5
OCF Endpoint information	"eps"	No	Defined in clause 7.8.2.5.6
OCF Interface	"if"	Yes	Defined in clause 7.6
Link instance	"ins"	No	Defined in clause 7.8.2.5.2
Policy	"p"	No	Defined in clause 7.8.2.5.3
Resource Type	"rt"	Yes	Defined in clause 7.4
Media type	"type"	No	Defined in clause 7.8.2.5.4
Position description Semantic Tag	"tag-pos-desc"	No	Defined in clause 11.5.2.1.2
Relative position Semantic Tag	"tag-pos-pos"	No	Defined in clause 11.5.2.1.3
Function description Semantic Tag	"tag-func-desc"	No	Defined in clause 11.5.2.2.2

1756 Note: Other Link target attributes may to defined for features in other specifications and may not be included in this table.

1757 **7.8.2.5.2 "ins" or Link instance Parameter**

1758 The "ins" Parameter identifies a particular Link instance in a list of Links. The "ins" Parameter may
 1759 be used to modify or delete a specific Link in a list of Links. The value of the "ins" Parameter is set
 1760 at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has
 1761 been set, the "ins" Parameter shall not be modified for as long as the Link is a member of that list.

1762 **7.8.2.5.3 "p" or policy Parameter**

1763 The policy Parameter defines various rules for correctly accessing a Resource referenced by a
 1764 target URI. The policy rules are configured by a set of key-value pairs.

1765 The policy Parameter "p" is defined by:

- 1766 – "bm" key: The "bm" key corresponds to an integer value that is interpreted as an 8-bit bitmask.
1767 Each bit in the bitmask corresponds to a specific policy rule. The rules are specified for "bm" in
1768 Table 11:

1769

Table 11 – "bm" Property definition

Bit Position	Policy rule	Comment
Bit 0 (the LSB)	discoverable	The discoverable rule defines whether the Link is to be included in the Resource discovery message via "/oic/res". If the Link is to be included in the Resource discovery message, then "p" shall include the "bm" key and set the discoverable bit to value 1. If the Link is NOT to be included in the Resource discovery message, then "p" shall either include the "bm" key and set the discoverable bit to value 0 or omit the "bm" key entirely.
Bit 1 (2 nd LSB)	observable	The Observable rule defines whether the Resource referenced by the target URI supports the NOTIFY operation. With the self-link, i.e. the Link with "rel" value of "self", "/oic/res" can have a Link with the target URI of "/oic/res" and indicate itself Observable. The "self" is defined by IETF RFC 4287 and registered in the IANA Registry for "rel" value defined at IANA Link Relations. If the Resource supports the NOTIFY operation, then "p" shall include the "bm" key and set the Observable bit to value 1. If the Resource does NOT support the NOTIFY operation, then "p" shall either include the "bm" key and set the Observable bit to value 0 or omit the "bm" key entirely.
Bits 2-7	--	Reserved for future use. All reserved bits in "bm" shall be set to value 0.

1770

1771 NOTE If all the bits in "bm" are defined to value 0, then the "bm" key may be omitted entirely from "p" as an efficiency
1772 measure. However, if any bit is set to value 1, then "bm" shall be included in "p" and all the bits shall be defined
1773 appropriately.

- 1774 – In a payload sent in response to a request that includes an OCF-Accept-Content-Format-
1775 Version option the "eps" Parameter shall provide the information for an encrypted connection.
- 1776 – Note that access to the Resource is controlled by the ACL for the Resource. A successful
1777 encrypted connection does not ensure that the requested action will succeed. See
1778 ISO/IEC 30118-2:2018 clause 12 for more information.

1779 This shows the policy Parameter for a Resource that is discoverable but not Observable.

```
1780 "p": {"bm": 1}
```

1781 This shows a self-link, i.e. the "/oic/res" Link in itself that is discoverable and Observable.

```
1782 {  
1783   "href": "/oic/res",  
1784   "rel": "self",  
1785   "rt": ["oic.wk.res"],  
1786   "if": ["oic.if.ll", "oic.if.baseline"],  
1787   "p": {"bm": 3}  
1788 }
```

1789 **7.8.2.5.4 "type" or media type Parameter**

1790 The "type" Parameter may be used to specify the various media types that are supported by a
1791 specific target Resource. The default type of "application/vnd.ocf+cbor" shall be used when the

1792 "type" element is omitted. Once a Client discovers this information for each Resource, it may use
1793 one of the available representations in the appropriate header field of the Request or Response.

1794 **7.8.2.5.5 "di" or Device ID Parameter**

1795 The "di" Parameter specifies the Device ID of the Device that hosts the target Resource defined in
1796 the in the "href" Parameter.

1797 The Device ID may be used to qualify a relative reference used in the "href" or to lookup OCF
1798 Endpoint information for the relative reference.

1799 **7.8.2.5.6 "eps" Parameter**

1800 The "eps" Parameter indicates the OCF Endpoint information of the target Resource.

1801 "eps" shall have as its value an array of items and each item represents OCF Endpoint information
1802 with "ep" and "pri" as specified in 10.2. "ep" is mandatory but "pri" is optional.

1803 This is an example of "eps" with multiple OCF Endpoints.

```
1804 "eps": [  
1805   { "ep": "coap://[fe80::b1d6]:1111", "pri": 2},  
1806   { "ep": "coaps://[fe80::b1d6]:1122"},  
1807   { "ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}  
1808 ]
```

1809 When "eps" is present in a link, the OCF Endpoint information in "eps" can be used to access the
1810 target Resource referred by the "href" Parameter.

1811 Note that the type of OCF Endpoint – Secure or Unsecure – that a Resource exposes merely
1812 determines the connection type(s) guaranteed to be available for sending requests to the Resource.
1813 For example, if a Resource only exposes a single CoAP "ep", it does not guarantee that the
1814 Resource cannot also be accessed via a Secure OCF Endpoint (e.g. via a CoAPS "ep" from another
1815 Resource's "eps information). Nor does exposing a given type of OCF Endpoint ensure that access
1816 to the Resource will be granted using the "ep" information. Whether requests to the Resource are
1817 granted or denied by the Access Control layer is separate from the "eps" information, and is
1818 determined by the configuration of the /acl2 Resource (see ISO/IEC 30118-2:2018 clause 13.5.3
1819 for details).

1820 When present, max-age information (e.g. Max-Age option for CoAP defined in IETF RFC 7252)
1821 determines the maximum time "eps" values may be cached before they are considered stale.

1822 **7.8.2.6 Formatting**

1823 When formatting in JSON, the list of Links shall be an array.

1824 **7.8.2.7 List of Links in a Collection**

1825 A Resource that exposes one or more Properties that are defined to be an array of Links where
1826 each Link can be discretely accessed is a Collection. The Property Name "links" is recommended
1827 for such an array of Links.

1828 This is an example of a Resource with a list of Links.

```
1829 /Room1  
1830 {  
1831   "rt": ["oic.wk.col"],  
1832   "if": ["oic.if.ll", "oic.if.baseline" ],  
1833   "color": "blue",  
1834   "links":  
1835   [  
1836     {
```

```

1837     "href": "/switch",
1838     "rt": ["oic.r.switch.binary"],
1839     "if": [ "oic.if.a", "oic.if.baseline" ],
1840     "p": {"bm": 3}
1841   },
1842   {
1843     "href": "/brightness",
1844     "rt": ["oic.r.light.brightness"],
1845     "if": [ "oic.if.a", "oic.if.baseline" ],
1846     "p": {"bm": 3}
1847   }
1848 ]
1849 }

```

1850 **7.8.2.8 Properties describing an array of Links**

1851 If a Resource Type that defines an array of Links (e.g. Collections, Atomic Measurements) has
 1852 restrictions on the "rt" values that can be within the array of Links, the Resource Type will define
 1853 the "rts" Property. The "rts" Property as defined in Table 12 will include all "rt" values allowed for
 1854 all Links in the array. If the Resource Type does not define the "rts" Property or the "rts" Property
 1855 is an empty array, then any "rt" value is permitted in the array of Links.

1856 For all instances of a Resource Type that defines the "rts" Property, the "rt" Link Parameter in
 1857 every Link in the array of Links shall be one of the "rt" values that is included in the "rts"
 1858 Property.

1859 **Table 12 – Resource Types Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Types	"rts"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	No	An array of Resource Types that are supported within an array of Links exposed by a Resource.

1860

1861 If a Resource Type that defines an array of Links has "rt" values which are required to be in the
 1862 array, the Resource Type will define the "rts-m" Property, as defined in Table 13, which will contain
 1863 all of the "rt" values that are required to be in the array of Links. If "rts-m" is defined, and "rts" is
 1864 defined and is not an empty array, then the "rt" values present in "rts-m" will be part of the values
 1865 present in "rts". Moreover, if the "rts-m" Property is defined, it shall be mandated (i.e. included in
 1866 the "required" field of a JSON definition) in the Resource definition and Introspection Device Data
 1867 (see 11.4).

1868 For all instances of a Resource Type that defines the "rts-m" Property, there shall be at least one
 1869 Link in the array of Links corresponding to each one of the "rt" values in the "rts-m" Property; for
 1870 all such Links the "rt" Link Parameter shall contain at least one of the "rt" values in the "rts-m"
 1871 Property.

1872 **Table 13 – Mandatory Resource Types Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Mandatory Resource Types	"rts-m"	"array"	Array of strings, conveying Resource Type IDs	N/A	R	No	An array of Resource Types that are mandatory to be exposed within an array of Links exposed by a Resource.

1873

1874 7.8.3 Collections

1875 7.8.3.1 Overview

1876 A Resource that contains one or more references (specified as Links) to other Resources is a
1877 Collection. These references may be related to each other or just be a list; the Collection provides
1878 a means to refer to this set of references with a single handle (i.e. the URI). A simple Resource is
1879 kept distinct from a Collection. Any Resource may be turned into a Collection by binding Resource
1880 references as Links. Collections may be used for creating, defining or specifying hierarchies,
1881 indexes, groups, and so on.

1882 A Collection shall have at least one Resource Type and at least one OCF Interface bound at all
1883 times during its lifetime. During creation time of a Collection the Resource Type and OCF Interfaces
1884 are specified. The initial defined Resource Types and OCF Interfaces may be updated during its
1885 life time. These initial values may be overridden using mechanism used for overriding in the case
1886 of a Resource. Additional Resource Types and OCF Interfaces may be bound to the Collection at
1887 creation or later during the lifecycle of the Collection.

1888 A Collection shall define a Property that is an array with zero or more Links. The target URIs in the
1889 Links may reference another Collection or another Resource. The referenced Collection or
1890 Resource may reside on the same Device as the Collection that includes that Link (called a local
1891 reference) or may reside on another Device (called a remote reference). The context URI of the
1892 Links in the array shall (implicitly) be the Collection that contains that Property. The (implicit)
1893 context URI may be overridden with explicit specification of the "anchor" Parameter in the Link
1894 where the value of "anchor" is the new base of the Link.

1895 A Resource may be referenced in more than one Collection, therefore, a unique parent-child
1896 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the
1897 Resource referenced in the Collection, i.e., the application may use Collections to represent a
1898 relationship but none is automatically implied or defined. The lifecycles of the Collection and the
1899 referenced Resource are also independent of one another.

1900 In the following example a Property "links" represents the list of Links in a Collection. The "links"
1901 Property has, as its value, an array of items and each item is a Link.

```
1902 /my/house ← This is IRI/URI of the Resource
1903 {
1904   "rt": ["my.r.house"], ← This and the next 3 lines are the Properties of the
1905 Resource.
1906   "color": "blue",
1907   "n": "myhouse",
1908   "links": [
1909     { ← This and the next 4 lines are the Parameters of a Link
1910       "href": "/door",
1911       "rt": ["oic.r.door"],
1912       "if": ["oic.if.a", "oic.if.baseline"]
1913     },
1914
1915     {
1916       "href": "/door/lock.status",
1917       "rt": ["oic.r.lock"],
1918       "if": ["oic.if.a", "oic.if.baseline"]
1919     },
1920
1921     {
1922       "href": "/light",
1923       "rt": ["oic.r.light"],
1924       "if": ["oic.if.s", "oic.if.baseline"]
```

```

1925     },
1926
1927     {
1928         "href": "/binarySwitch",
1929         "rt": ["oic.r.switch.binary"],
1930         "if": ["oic.if.a", "oic.if.baseline"]
1931     }
1932
1933     ]
1934 }

```

1935 A Collection may be:

- 1936 – A pre-defined Collection where the Collection has been defined a priori and the Collection is
1937 static over its lifetime. Such Collections may be used to model, for example, an appliance that
1938 is composed of other Devices or fixed set of Resources representing fixed functions.
- 1939 – A Device local Collection where the Collection is used only on the Device that hosts the
1940 Collection. Such Collections may be used as a short-hand on a Client for referring to many
1941 Servers as one.
- 1942 – A centralized Collection where the Collection is hosted on a Device but other Devices may
1943 access or update the Collection.
- 1944 – A hosted Collection where the Collection is centralized but is managed by an authorized agent
1945 or party.

1946 7.8.3.2 Collection Properties

1947 A Collection shall define a Property that is an array of Links (the Property Name "links" is
1948 recommended). In addition, other Properties may be defined for the Collection by the Resource
1949 Type. The mandatory and recommended Common Properties for a Collection are shown in Table 14.
1950 This list of Common Properties is in addition to those defined for Resources in 7.3.2.

1951 **Table 14 – Common Properties for Collections (in addition to Common Properties defined**
1952 **in 7.3.2)**

Property	Description	Property Name	Value Type	Mandatory
Links	The array of Links in the Collection	Per Resource Type definition	json Array of Links	Yes
Resource Types	The list of allowed Resource Types for Links in the Collection. If this Property is not defined or is null string then any Resource Type is permitted	As defined in Table 12	As defined in Table 12	No
Mandatory Resource Types	The list of Resource Types for Links that are mandatory in the Collection.	As defined in Table 13	As defined in Table 13	No

1953

1954 7.8.3.3 Default Resource Type

1955 A default Resource Type, "oic.wk.col", is available for Collections. This Resource Type shall be
1956 used only when another type has not been defined on the Collection or when no Resource Type
1957 has been specified at the creation of the Collection.

1958 The default Resource Type provides support for the Common Properties including an array of Links
 1959 with the Property Name "links".

1960 **7.8.3.4 Default OCF Interface**

1961 All instances of a Collection shall support the links list ("oic.if.ll") OCF Interface in addition to the
 1962 baseline ("oic.if.baseline") OCF Interface. An instance of a Collection may optionally support
 1963 additional OCF Interfaces that are defined within this document. The Default OCF Interface for a
 1964 Collection shall be links list ("oic.if.ll") unless otherwise specified by the Resource Type definition.

1965 **7.8.4 Atomic Measurement**

1966 **7.8.4.1 Overview**

1967 Certain use cases require that the Properties of multiple Resources are only accessible as a group
 1968 and individual access to those Properties of each Resource by a Client is prohibited. The Atomic
 1969 Measurement Resource Type is defined to meet this requirement. This is accomplished through
 1970 the use of the Batch OCF Interface.

1971 **7.8.4.2 Atomic Measurement Properties**

1972 An Atomic Measurement shall define a Property that is an array of Links (the Property Name "links"
 1973 is recommended). In addition, other Properties may be defined for the Atomic Measurement by the
 1974 Resource Type. The mandatory and recommended Common Properties for an Atomic
 1975 Measurement are shown in Table 15. This list of Common Properties is in addition to those defined
 1976 for Resources in 7.3.2.

1977 **Table 15 – Common Properties for Atomic Measurement (in addition to Common Properties**
 1978 **defined in 7.3.2)**

Property	Description	Property Name	Value Type	Mandatory
Links	The array of Links in the Atomic Measurement	Per Resource Type definition	json Array of Links	Yes
Resource Types	The list of allowed Resource Types for Links in the Atomic Measurement. If this Property is not defined or is null string then any Resource Type is permitted	As defined in Table 12	As defined in Table 12	No
Mandatory Resource Types	The list of Resource Types for Links that are mandatory in the Atomic Measurement.	As defined in Table 13	As defined in Table 13	No

1979

1980 **7.8.4.3 Normative behaviour**

1981 The normative behaviour of an Atomic Measurement is as follows:

- 1982 – The behaviour of the Batch OCF Interface ("oic.if.b") on the Atomic Measurement is defined as
 1983 follows:
- 1984 – Only RETRIEVE and NOTIFY operations are supported, for Batch OCF Interface, on Atomic
 1985 Measurement; the behavior of the RETRIEVE and NOTIFY operations shall be the same as
 1986 specified in 7.6.3.4, with exceptions as provided for in 7.8.4.3.

- 1987 – The UPDATE operation is not allowed, for Batch OCF Interface, on Atomic Measurement; if
- 1988 an UPDATE operation is received, it shall result in a method not allowed error code.
- 1989 – An error response shall not include any representation of a linked Resource (i.e. empty
- 1990 response for all linked Resources).
- 1991 – Any linked Resource within an Atomic Measurement (i.e. the target Resource of a Link in an
- 1992 Atomic Measurement) is subject to the following conditions:
- 1993 – Linked Resources within an Atomic Measurement and the Atomic Measurement itself shall
- 1994 exist on a single Server.
- 1995 – CRUDN operations shall not be allowed on linked Resources and shall result in a forbidden
- 1996 error code.
- 1997 – Linked Resources shall not expose the "oic.if.ll" OCF Interface. Since CRUDN operations
- 1998 are not allowed on linked Resources, the "oic.if.ll" OCF Interface would never be accessible.
- 1999 – Links to linked Resources in an Atomic Measurement shall only be accessible through the
- 2000 "oic.if.ll" or the "oic.if.baseline" OCF Interfaces of an Atomic Measurement.
- 2001 – The linked Resources shall not be listed in "/oic/res".
- 2002 – A linked Resource in an Atomic Measurement shall have defined one of "oic.if.a", "oic.if.s",
- 2003 "oic.if.r", or "oic.if.rw" as its Default OCF Interface.
- 2004 – Not all linked Resources in an Atomic Measurement are required to be Observable. If an Atomic
- 2005 Measurement is being Observed using the "oic.if.b" OCF Interface, notification responses shall
- 2006 not be generated when the linked Resources which are not marked Observable are updated or
- 2007 change state.
- 2008 – All linked Resources in an Atomic Measurement shall be included in every RETRIEVE and
- 2009 Observe response when using the "oic.if.b" OCF Interface.
- 2010 – An Atomic Measurement shall support the "oic.if.b" and the "oic.if.ll" OCF Interfaces.
- 2011 – Filtering of linked Resources in an Atomic Measurement is not allowed. Query parameters that
- 2012 select one or more individual linked Resources in a request to an Atomic Measurement shall
- 2013 result in a "forbidden" error code.
- 2014 – If the "rel" Link Parameter is included in a Link contained in an Atomic Measurement, it shall
- 2015 have either the "hosts" or the "item" value.
- 2016 – The Default OCF Interface of an Atomic Measurement is "oic.if.b".

2017 **7.8.4.4 Security considerations**

2018 Access rights to an Atomic Measurement Resource Type is as specified in clause 12.2.7.2 (ACL

2019 considerations for batch request to the Atomic Measurement Resource Type) of ISO/IEC 30118-

2020 2:2018).

2021 **7.8.4.5 Default Resource Type**

2022 The Resource Type is defined as "oic.wk.atomicmeasurement" as defined in Table 16.

2023 **Table 16 – Atomic Measurement Resource Type**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction	M/CR/O
none	Atomic Measurement	"oic.wk.atomicmeasurement"	"oic.if.ll" "oic.if.baseline" "oic.if.b"	A specialisation of the Collection pattern to ensure atomic RETRIEVAL of its referred Resources	RETRIEVE, NOTIFY	O

2024

2025 The Properties for Atomic Measurement are as defined in Table 17.

2026 **Table 17 – Properties for Atomic Measurement (in addition to Common Properties defined**
2027 **in 7.3.2)**

Property	Description	Property name	Value Type	Mandatory
Links	The set of links that point to the linked Resources	Per Resource Type definition	json Array of Links	Yes

2028

2029 **7.9 Query Parameters**

2030 **7.9.1 Introduction**

2031 Properties and Parameters (including those that are part of a Link) may be used in the query part
2032 of a URI (see 6.2.2) as one criterion for selection of a particular Resource. This is done by declaring
2033 the Property (i.e. <Property Name> = <desired Property Value>) as one of the segments of the
2034 query. Only ASCII strings are permitted in query filters, and NULL characters are disallowed in
2035 query filters. This means that only Property Values with ASCII characters may be matched in a
2036 query filter.

2037 The Resource is selected when all the declared Properties or Link Parameters in the query match
2038 the corresponding Properties or Link Parameters in the target.

2039 **7.9.2 Use of multiple parameters within a query**

2040 When a query contains multiple separate query parameters these are delimited by an "&" as
2041 described in 6.2.2.

2042 A Client may apply multiple separate query parameters, for
2043 example "?ins=11111&rt=oic.r.switch.binary". If such queries are supported by the Server this shall
2044 be accomplished by matching "all of" the different query parameter types ("rt", "ins", "if", etc)
2045 against the target of the query. In the example, this resolves to an instance of oic.r.switch.binary
2046 that also has an "ins" populated as "11111". There is no significance applied to the order of the
2047 query parameters.

2048 A Client may select more than one Resource Type using repeated query parameters, for example
2049 "?rt=oic.r.switch.binary&rt=oic.r.ramptime". If such queries are supported by the Server this shall
2050 be accomplished by matching "any of" the repeated query parameters against the target of the
2051 query. In the example, any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that may exist
2052 are selected.

2053 A Client may combine both multiple repeated parameters and multiple separate parameters in a
2054 single query, for example "?if=oic.if.b&ins=11111&rt=oic.r.switch.binary&rt=oic.r.ramptime". If
2055 such queries are supported by the Server this shall be accomplished by matching "any of" the
2056 repeated query parameters and then matching "all of" the different query parameter types. In the
2057 example any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that also have an "ins" of
2058 "11111" that may exist are selected in a batch response.

2059 NOTE The parameters within a query string are represented within the actual messaging protocol as defined in clause
2060 11.5.

2061 **7.9.3 Application to multi-value "rt" Resources**

2062 An "rt" query for a multi-value "rt" Resource with the Default OCF Interface of "oic.if.a", "oic.if.s",
2063 "oic.if.r", "oic.if.rw" or "oic.if.baseline" is an extension of a generic "rt" query. When a Server
2064 receives a RETRIEVE request for a multi-value "rt" Resource with an "rt" query, (i.e. GET
2065 /ResExample?rt=oic.r.foo), the Server should respond only when the query value is an item of the

2066 "rt" Property Value of the target Resource and should send back only the Properties associated
 2067 with the query value(s). For example, upon receiving GET /ResExample?rt=oic.r.switch.binary
 2068 targeting a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"], the Server responds
 2069 with only the Properties of oic.r.switch.binary.

2070 **7.9.4 OCF Interface specific considerations for queries**

2071 **7.9.4.1 OCF Interface selection**

2072 When an OCF Interface is to be selected for a request, it shall be specified as a query parameter
 2073 in the URI of the Resource in the request message. If no query parameter is specified, then the
 2074 Default OCF Interface shall be used. If the selected OCF Interface is not one of the permitted OCF
 2075 Interfaces on the Resource then selecting that OCF Interface is an error and the Server shall
 2076 respond with an error response code.

2077 For example, the baseline OCF Interface may be selected by adding "if=oic.if.baseline" to the list
 2078 of query parameters in the URI of the target Resource. For example: "GET
 2079 /oic/res?if=oic.if.baseline".

2080 **7.9.4.2 Batch OCF Interface**

2081 See 7.6.3.4 for details on the batch OCF Interface itself. Query parameters may be used with the
 2082 batch OCF Interface in order to select particular Resources in a Collection for retrieval or update;
 2083 these parameters are used to select items in the Collection by matching Link Parameter Values.

2084 When Link selection query parameters are used with RETRIEVE operations applied using the batch
 2085 OCF Interface, only the Resources in the Collection with matching Link Parameters should be
 2086 returned.

2087 When Link selection query parameters are used with UPDATE operations applied using the batch
 2088 OCF Interface, only the Resources having matching Link Parameters should be updated.

2089 See 7.6.3.4.5 for examples of RETRIEVE and UPDATE operations that use Link selection query
 2090 parameters.

2091 **8 CRUDN**

2092 **8.1 Overview**

2093 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for
 2094 manipulating Resources. These operations are performed by a Client on the Resources contained
 2095 in n Server.

2096 On reception of a valid CRUDN operation a Server hosting the Resource that is the target of the
 2097 request shall generate a response depending on the OCF Interface included in the request; or
 2098 based on the Default OCF Interface for the Resource Type if no OCF Interface is included.

2099 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in
 2100 Table 18. A Device shall use CBOR as the default payload (content) encoding scheme for Resource
 2101 representations included in CRUDN operations and operation responses; a Device may negotiate
 2102 a different payload encoding scheme (e.g, see in 12.2.4 for CoAP messaging). Clauses 8.2 through
 2103 8.6 respectively specify the CRUDN operations and use of the parameters. The type definitions for
 2104 these terms will be mapped in the clause 11.5 for each protocol.

2105 **Table 18 – Parameters of CRUDN messages**

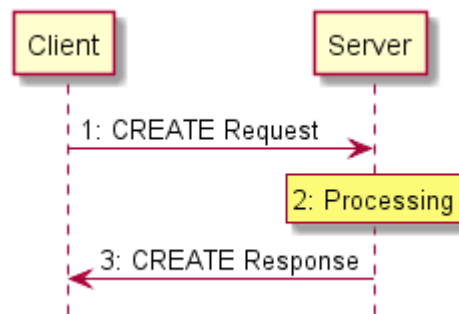
Applicability	Name	Denotation	Definition
All messages	<i>fr</i>	From	The URI of the message originator.

	<i>to</i>	To	The URI of the recipient of the message.
	<i>ri</i>	Request Identifier	The identifier that uniquely identifies the message in the originator and the recipient.
	<i>cn</i>	Content	Information specific to the operation.
Requests	<i>op</i>	Operation	Specific operation requested to be performed by the Server.
	<i>obs</i>	Observe	Indicator for an Observe request.
Responses	<i>rs</i>	Response Code	Indicator of the result of the request; whether it was accepted and what the conclusion of the operation was. The values of the response code for CRUDN operations shall conform to those as defined in clause 5.9 and 12.1.2 in IETF RFC 7252.
	<i>obs</i>	Observe	Indicator for an Observe response.

2106 **8.2 CREATE**

2107 **8.2.1 Overview**

2108 The CREATE operation is used to request the creation of new Resources on the Server. The
 2109 CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 5.



2110

2111 **Figure 5 – CREATE operation**

2112 **8.2.2 CREATE request**

2113 The CREATE request message is transmitted by the Client to the Server to create a new Resource
 2114 by the Server. The CREATE request message will carry the following parameters:

- 2115 – *fr*: Unique identifier of the Client
- 2116 – *to*: URI of the target Resource responsible for creation of the new Resource.
- 2117 – *ri*: Identifier of the CREATE request.
- 2118 – *cn*: Information of the Resource to be created by the Server.
- 2119 – *cn* will include the URI and Resource Type Property of the Resource to be created.
- 2120 – *cn* may include additional Properties of the Resource to be created.
- 2121 – *op*: CREATE

2122 **8.2.3 Processing by the Server**

2123 Following the receipt of a CREATE request, the Server may validate if the Client has the
 2124 appropriate rights for creating the requested Resource. If the validation is successful, the Server
 2125 creates the requested Resource. The Server caches the value of *ri* parameter in the CREATE
 2126 request for inclusion in the CREATE response message.

2127 **8.2.4 CREATE response**

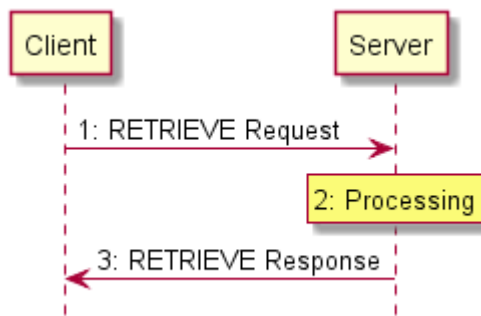
2128 The Server shall transmit a CREATE response message in response to a CREATE request
2129 message from a Client. The CREATE response message will include the following parameters:

- 2130 – *fr*: Unique identifier of the Server
- 2131 – *to*: Unique identifier of the Client
- 2132 – *ri*: Identifier included in the CREATE request
- 2133 – *cn*: Information of the Resource as created by the Server.
 - 2134 – *cn* will include the URI of the created Resource.
 - 2135 – *cn* will include the Resource representation of the created Resource.
- 2136 – *rs*: The result of the CREATE operation.

2137 **8.3 RETRIEVE**

2138 **8.3.1 Overview**

2139 The RETRIEVE operation is used to request the current state or representation of a Resource. The
2140 RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in Figure 6.



2141

2142

Figure 6 – RETRIEVE operation

2143 **8.3.2 RETRIEVE request**

2144 RETRIEVE request message is transmitted by the Client to the Server to request the representation
2145 of a Resource from a Server. The RETRIEVE request message will carry the following parameters:

- 2146 – *fr*: Unique identifier of the Client.
- 2147 – *to*: URI of the Resource the Client is targeting.
- 2148 – *ri*: Identifier of the RETRIEVE request.
- 2149 – *op*: RETRIEVE.

2150 **8.3.3 Processing by the Server**

2151 Following the receipt of a RETRIEVE request, the Server may validate if the Client has the
2152 appropriate rights for retrieving the requested data and the Properties are readable. The Server
2153 caches the value of *ri* parameter in the RETRIEVE request for use in the response

2154 **8.3.4 RETRIEVE response**

2155 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
2156 message from a Client. The RETRIEVE response message will include the following parameters:

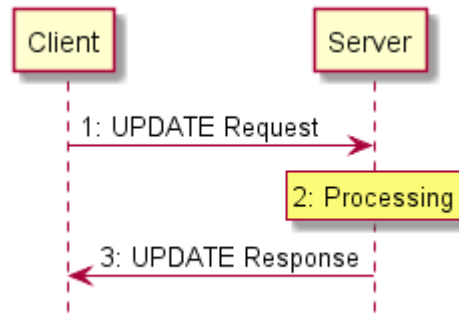
- 2157 – *fr*: Unique identifier of the Server.
- 2158 – *to*: Unique identifier of the Client.

- 2159 – *ri*: Identifier included in the RETRIEVE request.
- 2160 – *cn*: Information of the Resource as requested by the Client.
- 2161 – *cn* should include the URI of the Resource targeted in the RETRIEVE request.
- 2162 – *rs*: The result of the RETRIEVE operation.

2163 **8.4 UPDATE**

2164 **8.4.1 Overview**

2165 The UPDATE operation is either a Partial UPDATE or a complete replacement of the information
 2166 in a Resource in conjunction with the OCF Interface that is also applied to the operation. The
 2167 UPDATE operation is initiated by the Client and consists of three steps, as depicted in Figure 7.



2168

2169 **Figure 7 – UPDATE operation**

2170 **8.4.2 UPDATE request**

2171 The UPDATE request message is transmitted by the Client to the Server to request the update of
 2172 information of a Resource on the Server. The UPDATE request message will carry the following
 2173 parameters:

- 2174 – *fr*: Unique identifier of the Client.
- 2175 – *to*: URI of the Resource targeted for the information update.
- 2176 – *ri*: Identifier of the UPDATE request.
- 2177 – *op*: UPDATE.
- 2178 – *cn*: Information, including Properties, of the Resource to be updated at the target Resource.

2179 **8.4.3 Processing by the Server**

2180 **8.4.3.1 Overview**

2181 Following the receipt of an UPDATE request, the Server may validate if the Client has the
 2182 appropriate rights for updating the requested data. If the validation is successful the Server updates
 2183 the target Resource information according to the information carried in *cn* parameter of the
 2184 UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE request
 2185 for use in the response.

2186 An UPDATE request that includes Properties that are read-only shall be rejected by the Server with
 2187 an *rs* indicating a bad request.

2188 An UPDATE request shall be applied only to the Properties in the target Resource visible via the
 2189 applied OCF Interface that support the operation. An UPDATE of non-existent Properties is ignored.

2190 An UPDATE request shall be applied to the Properties in the target Resource even if those Property
 2191 Values are the same as the values currently exposed by the target Resource.

2192 **8.4.3.2 Resource monitoring by the Server**

2193 The Server shall monitor the state the Resource identified in the Observe request from the Client.
2194 Anytime there is a change in the state of the Observed Resource or an UPDATE operation applied
2195 to the Resource, the Server sends another RETRIEVE response with the Observe indication. The
2196 mechanism does not allow the Client to specify any bounds or limits which trigger a notification,
2197 the decision is left entirely to the Server.

2198 **8.4.3.3 Additional RETRIEVE responses with Observe indication**

2199 The Server shall transmit updated RETRIEVE response messages following Observed changes in
2200 the state of the Resources requested by the Client. The RETRIEVE response message shall include
2201 the parameters listed in 11.3.2.4.

2202 **8.4.4 UPDATE response**

2203 The UPDATE response message will include the following parameters:

- 2204 – *fr*: Unique identifier of the Server.
- 2205 – *to*: Unique identifier of the Client.
- 2206 – *ri*: Identifier included in the UPDATE request.
- 2207 – *rs*: The result of the UPDATE request.

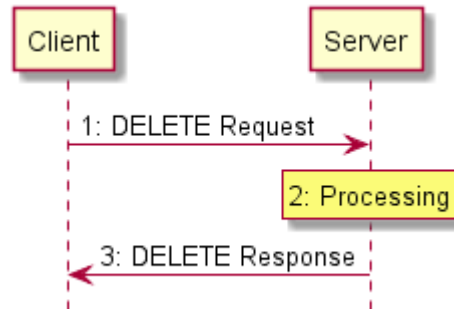
2208 The UPDATE response message may also include the following parameters:

- 2209 – *cn*: The Resource representation following processing of the UPDATE request.

2210 **8.5 DELETE**

2211 **8.5.1 Overview**

2212 The DELETE operation is used to request the removal of a Resource. The DELETE operation is
2213 initiated by the Client and consists of three steps, as depicted in Figure 8.



2214
2215 **Figure 8 – DELETE operation**

2216 **8.5.2 DELETE request**

2217 DELETE request message is transmitted by the Client to the Server to delete a Resource on the
2218 Server. The DELETE request message will carry the following parameters:

- 2219 – *fr*: Unique identifier of the Client.
- 2220 – *to*: URI of the target Resource which is the target of deletion.
- 2221 – *ri*: Identifier of the DELETE request.
- 2222 – *op*: DELETE.

2223 **8.5.3 Processing by the Server**

2224 Following the receipt of a DELETE request, the Server may validate if the Client has the appropriate
2225 rights for deleting the identified Resource, and whether the identified Resource exists. If the
2226 validation is successful, the Server removes the requested Resource and deletes all the associated
2227 information. The Server caches the value of *ri* parameter in the DELETE request for use in the
2228 response.

2229 **8.5.4 DELETE response**

2230 The Server shall transmit a DELETE response message in response to a DELETE request message
2231 from a Client. The DELETE response message will include the following parameters:

- 2232 – *fr*: Unique identifier of the Server.
- 2233 – *to*: Unique identifier of the Client.
- 2234 – *ri*: Identifier included in the DELETE request.
- 2235 – *rs*: The result of the DELETE operation.

2236 **8.6 NOTIFY**

2237 **8.6.1 Overview**

2238 The NOTIFY operation is used to request asynchronous notification of state changes. Complete
2239 description of the NOTIFY operation is provided in 11.3. The NOTIFY operation uses the
2240 NOTIFICATION response message which is defined here.

2241 **8.6.2 NOTIFICATION response**

2242 The NOTIFICATION response message is sent by a Server to notify the URLs identified by the
2243 Client of a state change. The NOTIFICATION response message carries the following parameters:

- 2244 – *fr*: Unique identifier of the Server.
- 2245 – *to*: URI of the Resource target of the NOTIFICATION message.
- 2246 – *ri*: Identifier included in the CREATE request.
- 2247 – *op*: NOTIFY.
- 2248 – *cn*: The updated state of the Resource.

2249 **9 Network and connectivity**

2250 **9.1 Introduction**

2251 The Internet of Things is comprised of a wide range of applications which sense and actuate the
2252 physical world with a broad spectrum of device and network capabilities: from battery powered
2253 nodes transmitting 100 bytes per day and able to last 10 years on a coin cell battery, to mains
2254 powered nodes able to maintain Megabit video streams. It is estimated that many 10s of billions of
2255 IoT devices will be deployed over the coming years.

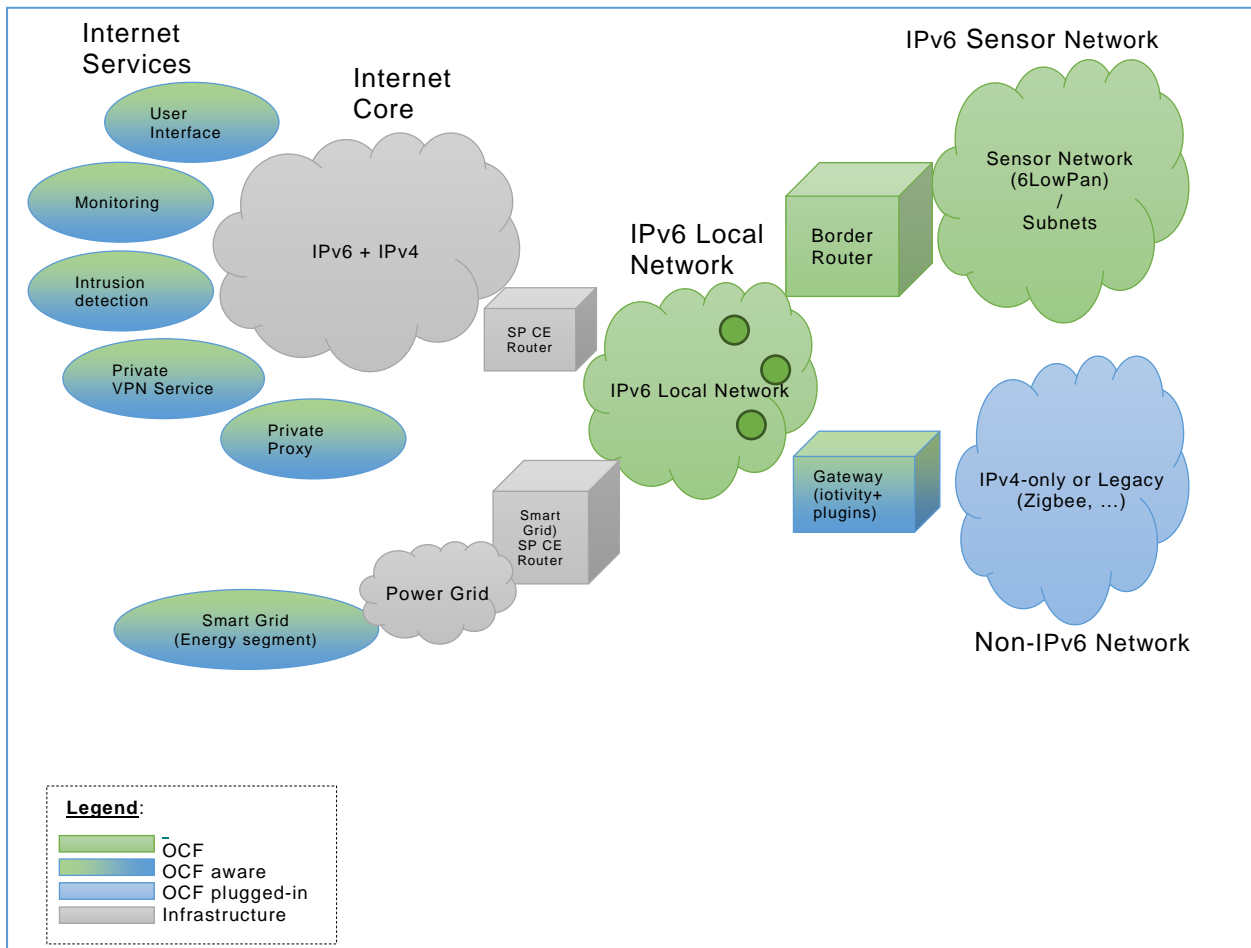
2256 It is desirable that the connectivity options be adapted to the IP layer. To that end, IETF has
2257 completed considerable work to adapt Bluetooth®, Wi-Fi, 802.15.4, LPWAN, etc. to IPv6. These
2258 adaptations, plus the larger address space and improved address management capabilities, make
2259 IPv6 the clear choice for the OCF network layer technology.

2260 **9.2 Architecture**

2261 While the aging IPv4 centric network has evolved to support complex topologies, its deployment
2262 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More
2263 complex network topologies, often seen in residential home, are mostly introduced through the
2264 acquisition of additional home network devices, which rely on technologies like private Network

2265 Address Translation (NAT). These technologies require expert assistance to set up correctly and
 2266 should be avoided in a home network as they most often result in breakage of constructs like
 2267 routing, naming and discovery services.

2268 The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices
 2269 and associated routers, but also new services introducing additional edge routers. All these new
 2270 requirements require advance architectural constructs to address complex network topologies like
 2271 the one shown in Figure 9.



2272

2273 **Figure 9 – High Level Network & Connectivity Architecture**

2274 In terms of IETF RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further
 2275 implement various specializations of those roles:

- 2276 – A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
- 2277 – Nodes limited in processing power, memory, non-volatile storage or transmission capacity
 2278 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).
 2279 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU
 2280 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, and Near Field Communication (NFC).
- 2281 – A node may translate and route messaging between IPv6 and non-IPv6 networks.

2282 **9.3 IPv6 network layer requirements**

2283 **9.3.1 Introduction**

2284 Projections indicate that many 10s of billions of new IoT endpoints and related services will be
2285 brought online in the next few years. These endpoint's capabilities will span from battery powered
2286 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating
2287 over Ethernet and WiFi links.

2288 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide
2289 variety of applications such as Web browsing, email, voice, video, and critical system monitoring
2290 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which
2291 is that available address space has been consumed.

2292 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF
2293 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 2294 – Larger address space. Side-effect: greatly reduce the need for NATs.
- 2295 – More flexible addressing architecture. Multiple addresses and types per interface: Link-local,
2296 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed
2297 networks, better re-numbering capability, etc.
- 2298 – More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 2299 – Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 2300 – All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
- 2301 – Major Service Providers around the globe are deploying IPv6.

2302 **9.3.2 IPv6 node requirements**

2303 **9.3.2.1 Introduction**

2304 In order to ensure network layer services interoperability from node to node, mandating a common
2305 network layer across all nodes is vital. The protocol should enable the network to be: secure,
2306 manageable, and scalable and to include constrained and self-organizing meshed nodes. OCF
2307 mandates IPv6 as the common network layer protocol to ensure interoperability across all Devices.
2308 More capable Devices may also include additional protocols creating multiple-stack Devices. The
2309 remainder of this clause will focus on interoperability requirements for IPv6 hosts, IPv6 constrained
2310 hosts and IPv6 routers. The various protocol translation permutations included in multi-stack
2311 gateway devices may be addresses in subsequent addendums of this document.

2312 **9.3.2.2 IP Layer**

2313 An IPv6 node shall support IPv6 and it shall conform to the requirements as specified in
2314 IETF RFC 6434.

2315 **10 OCF Endpoint**

2316 **10.1 OCF Endpoint definition**

2317 The specific definition of an OCF Endpoint depends on the Transport Protocol Suite being used.
2318 For the example of CoAP over UDP over IPv6, the OCF Endpoint is identified by an IPv6 address
2319 and UDP port number.

2320 Each Device shall associate with at least one OCF Endpoint with which it can exchange request
2321 and response messages. When a message is sent to an OCF Endpoint, it shall be delivered to the
2322 Device which is associated with the OCF Endpoint. When a request message is delivered to an
2323 OCF Endpoint, path component is enough to locate the target Resource.

2324 A Device can be associated with multiple OCF Endpoints. For example, n Device can have several
2325 IP addresses or port numbers or support both CoAP and HTTP transfer protocol. Different
Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 61

2326 Resources in n Device may be accessed with the same OCF Endpoint or need different ones. Some
2327 Resources may use one OCF Endpoint and others a different one. It depends on an implementation.

2328 On the other hand, an OCF Endpoint can be shared among multiple Devices, only when there is a
2329 way to clearly designate the target Resource with request URI. For example, when multiple CoAP
2330 servers use uniquely different URI paths for all their hosted Resources, and the CoAP
2331 implementation demultiplexes by path, they can share the same CoAP OCF Endpoint. However,
2332 this is not possible in this version of the document, because a pre-determined URI (e.g. "/oic/d") is
2333 mandatory for some mandatory Resources (e.g. "oic.wk.d").

2334 10.2 OCF Endpoint information

2335 10.2.1 Introduction

2336 OCF Endpoint is represented by OCF Endpoint information which consists of two items of key-
2337 value pair, "ep" and "pri".

2338 10.2.2 "ep"

2339 "ep" represents Transport Protocol Suite and OCF Endpoint Locator specified as follows:

2340 – *Transport Protocol Suite* - a combination of protocols (e.g. CoAP + UDP + IPv6) with which
2341 request and response messages can be exchanged for RESTful transaction (i.e. CRUDN). A
2342 Transport Protocol Suite shall be indicated by a URI scheme name. All scheme names
2343 supported by this document are IANA registered, these are listed in Table 19. A vendor may
2344 also make use of a non-IANA registered scheme name for their own use (e.g.
2345 "com.example.foo"), this shall follow the syntax for such scheme names defined by
2346 IETF RFC 7595. The behaviour of a vendor-defined scheme name is undefined by this
2347 document. All OCF defined Resource Types when exposing OCF Endpoint Information in an
2348 "eps" (see 10.2.4) shall include at least one "ep" with a Transport Protocol Suite as defined in
2349 Table 19.

2350 – *OCF Endpoint Locator* – an address (e.g. IPv6 address + Port number) or an indirect identifier
2351 (e.g., DNS name) resolvable to an IP address, through which a message can be sent to the
2352 OCF Endpoint and in turn associated Device. The OCF Endpoint Locator for "coap" and "coaps"
2353 shall be specified as "IP address: port number". The OCF Endpoint Locator for "coap+tcp" or
2354 "coaps+tcp" shall be specified as "IP address: port number" or "DNS name: port number" or
2355 "DNS name" such that the DNS name shall be resolved to a valid IP address for the target
2356 Resource with a name resolution service (i.e., DNS). For the 3rd case, when the port number
2357 is omitted, the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and for
2358 "coaps+tcp") scheme respectively as defined in IETF RFC 8323. Temporary addresses should
2359 not be used because OCF Endpoint Locators are for the purpose of accepting incoming
2360 sessions, whereas temporary addresses are for initiating outgoing sessions (IETF RFC 4941).
2361 Moreover, its inclusion in "/oic/res" can cause a privacy concern (IETF RFC 7721).

2362 "ep" shall have as its value a URI (as specified in IETF RFC 3986) with the scheme component
2363 indicating Transport Protocol Suite and the authority component indicating the OCF Endpoint
2364 Locator.

2365 An "ep" example for "coap" and "coaps" is as illustrated:

```
"ep": "coap://[fe80::b1d6]:1111"
```

2366 An "ep" example for "coap+tcp" and "coaps+tcp" is as illustrated:

```
"ep": "coap+tcp://[2001:db8:a::123]:2222"  
"ep": "coap+tcp://foo.bar.com:2222"  
"ep": "coap+tcp://foo.bar.com"
```


2367 The current list of "ep" with corresponding Transport Protocol Suite is shown in Table 19:

2368

Table 19 – "ep" value for Transport Protocol Suite

Transport Protocol Suite	scheme	OCF Endpoint Locator	"ep" Value example
coap+udp+ip	"coap"	IP address + port number	"coap://[fe80::b1d6]:1111"
coaps + udp + ip	"coaps"	IP address + port number	"coaps://[fe80::b1d6]:1122"
coap + tcp + ip	"coap+tcp"	IP address + port number DNS name: port number DNS name	"coap+tcp://[2001:db8:a::123]:2222" "coap+tcp://foo.bar.com:2222" "coap+tcp://foo.bar.com"
coaps + tcp + ip	"coaps+tcp"	IP address + port number DNS name: port number DNS name	"coaps+tcp://[2001:db8:a::123]:2233" "coaps+tcp://[2001:db8:a::123]:2233" "coaps+tcp://foo.bar.com:2233"

2369

2370 **10.2.3 "pri"**

2371 When there are multiple OCF Endpoints, "pri" indicates the priority among them.

2372 "pri" shall be represented as a positive integer (e.g. "pri": 1) and the lower the value, the higher the
2373 priority.

2374 The default "pri" value is 1, i.e. when "pri" is not present, it shall be equivalent to "pri": 1.

2375 **10.2.4 OCF Endpoint information in "eps" Parameter**

2376 To carry OCF Endpoint information, a new Link Parameter "eps" is defined in 7.8.2.5.6. "eps" has
2377 an array of items as its value and each item represents OCF Endpoint information with two key-
2378 value pairs, "ep" and "pri", of which "ep" is mandatory and "pri" is optional.

2379 OCF Endpoint Information in an "eps" Parameter is valid for the target Resource of the Link, i.e.,
2380 the Resource referred by "href" Parameter. OCF Endpoint information in an "eps" Parameter may
2381 be used to access other Resources on the Device, but such access is not guaranteed.

2382 A Client may resolve the "ep" value to an IP address for the target Resource, i.e., the address to
2383 access the Device which hosts the target Resource. A valid (transfer protocol) URI for the target
2384 Resource can be constructed with the scheme, host and port components from the "ep" value and
2385 the "path" component from the "href" value.

2386 Links with an "eps":

```
2387 {  
2388   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9 ",  
2389   "href": "/myLightSwitch",  
2390   "rt": ["oic.r.switch.binary"],  
2391   "if": ["oic.if.a", "oic.if.baseline"],  
2392   "p": {"bm": 3},  
2393   "eps": [  
2394     {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
```

```

2395     {"ep": "coaps://[fe80::b1d6]:1122"}
2396   ]
2397 }
2398
2399 {
2400   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2401   "href": "/myTemperature",
2402   "rt": ["oic.r.temperature"],
2403   "if": ["oic.if.a", "oic.if.baseline"],
2404   "p": {"bm": 3},
2405   "eps": [
2406     {"ep": "coap+tcp://foo.bar.com", "pri": 2},
2407     {"ep": "coaps+tcp://foo.bar.com:1122"}
2408   ]
2409 }

```

2410 In the previous example, "anchor" represents the hosting Device, "href", target Resource and "eps"
 2411 the two OCF Endpoints for the target Resource. The (fully-qualified) URIs for the target Resource
 2412 are as illustrated:

```

2413 coap://[fe80::b1d6]:1111/myLightSwitch
2414 coaps://[fe80::b1d6]:1122/myLightSwitch
2415 coap+tcp://foo.bar.com:5683/myTemperature

```

2416 coaps+tcp://foo.bar.com:1122/myTemperature If the target Resource of a Link requires a secure
 2417 connection (e.g. CoAPS), "eps" Parameter shall be used to indicate the necessary information (e.g.
 2418 port number) in OCF 1.0 payload. For optional backward compatibility with OIC 1.1, the "sec" and
 2419 "port" shall only be used in OIC 1.1 payload.

2420 **10.3 OCF Endpoint discovery**

2421 **10.3.1 Introduction**

2422 OCF Endpoint discovery is defined as the process for a Client to acquire the OCF Endpoint
 2423 information for Device or Resource.

2424 **10.3.2 Implicit discovery**

2425 If a Device is the source of a CoAP message (e.g. "/oic/res" response), the source IP address and
 2426 port number may be combined to form the OCF Endpoint Locator for the Device. Along with a
 2427 "coap" scheme and default "pri" value, OCF Endpoint information for the Device may be constructed.

2428 In other words, a "/oic/res" response message with CoAP may implicitly carry the OCF Endpoint
 2429 information of the responding Device and in turn all the hosted Resources, which may be accessed
 2430 with the same transfer protocol of CoAP. In the absence of an "eps" Parameter, a Client shall be
 2431 able to utilize implicit discovery to access the target Resource.

2432 **10.3.3 Explicit discovery with "/oic/res" response**

2433 OCF Endpoint information may be explicitly indicated with the "eps" Parameter of the Links in
 2434 "/oic/res".

2435 As in 10.3.2, an "/oic/res" response may implicitly indicate the OCF Endpoint information for some
 2436 Resources hosted by the responding Device. However implicit discovery, i.e., inference of OCF
 2437 Endpoint information from CoAP response message, may not work for some Resources on the
 2438 same Device. For example, some Resources may allow only secure access via CoAPS which
 2439 requires the "eps" Parameter to indicate the port number. Moreover "/oic/res" may expose a target
 2440 Resource which belongs to another Device.

2441 When the OCF Endpoint for a target Resource of a Link cannot be implicitly inferred, the "eps"
 2442 Parameter shall be included to provide explicit OCF Endpoint information with which a Client can
 2443 access the target Resource. In the presence of the "eps" Parameter, a Client shall be able to utilize

2444 it to access the target Resource. For "coap" and "coaps", a Client may use the IP address in the
2445 "ep" value in the "eps" Parameter to access the target Resource. For "coap+tcp" and "coaps+tcp",
2446 a Client may use the IP address in the "eps" Parameter or resolve the DNS name in the "eps"
2447 Parameter to acquire a valid IP address for the target Resource. If "eps" Parameter omits the port
2448 number, then the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and
2449 "coaps+tcp") scheme as defined in IETF RFC 8323. To access the target Resource of a Link, a
2450 Client may use the "eps" Parameter in the Link, if it is present and fall back on implicit discovery if
2451 not.

2452 This is an example of an "/oic/res" response from a Device having the "eps" Parameter in Links.

```
2453 [
2454   {
2455     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2456     "href": "/oic/res",
2457     "rel": "self",
2458     "rt": ["oic.wk.res"],
2459     "if": ["oic.if.ll", "oic.if.baseline"],
2460     "p": {"bm": 3},
2461     "eps": [
2462       {"ep": "coap://[2001:db8:a::b1d4]:5555"},
2463       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2464     ]
2465   },
2466   {
2467     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2468     "href": "/oic/d",
2469     "rt": ["oic.wk.d"],
2470     "if": ["oic.if.r", "oic.if.baseline"],
2471     "p": {"bm": 3},
2472     "eps": [
2473       {"ep": "coap://[2001:db8:a::b1d4]:5555"},
2474       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2475     ]
2476   },
2477   {
2478     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2479     "href": "/oic/p",
2480     "rt": ["oic.wk.p"],
2481     "if": ["oic.if.r", "oic.if.baseline"],
2482     "p": {"bm": 3},
2483     "eps": [
2484       {"ep": "coap://[2001:db8:a::b1d4]:5555"},
2485       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2486     ]
2487   },
2488   {
2489     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2490     "href": "/oic/sec/doxm",
2491     "rt": ["oic.r.doxm"],
2492     "if": ["oic.if.baseline"],
2493     "p": {"bm": 1},
2494     "eps": [
2495       {"ep": "coap://[2001:db8:a::b1d4]:5555"},
2496       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2497     ]
2498   },
2499   {
2500     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2501     "href": "/oic/sec/pstat",
2502     "rt": ["oic.r.pstat"],
2503
```

```

2504     "if": ["oic.if.baseline"],
2505     "p": {"bm": 1},
2506     "eps": [
2507         {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2508     ]
2509 },
2510 {
2511     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2512     "href": "/oic/sec/cred",
2513     "rt": ["oic.r.cred"],
2514     "if": ["oic.if.baseline"],
2515     "p": {"bm": 1},
2516     "eps": [
2517         {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2518     ]
2519 },
2520 {
2521     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2522     "href": "/oic/sec/acl2",
2523     "rt": ["oic.r.acl2"],
2524     "if": ["oic.if.baseline"],
2525     "p": {"bm": 1},
2526     "eps": [
2527         {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2528     ]
2529 },
2530 {
2531     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2532     "href": "/myIntrospection",
2533     "rt": ["oic.wk.introspection"],
2534     "if": ["oic.if.r", "oic.if.baseline"],
2535     "p": {"bm": 3},
2536     "eps": [
2537         {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2538     ]
2539 },
2540 {
2541     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2542     "href": "/myLight",
2543     "rt": ["oic.r.switch.binary"],
2544     "if": ["oic.if.a", "oic.if.baseline"],
2545     "p": {"bm": 3},
2546     "eps": [
2547         {"ep": "coaps://[2001:db8:a::b1d4]:22222"}
2548     ]
2549 }
2550 ]
2551

```

2552 The exact format of the "/oic/res" response and a way for a Client to acquire a "/oic/res" response
2553 message is specified in Annex A and 11.2.4 respectively.

2554 11 Functional interactions

2555 11.1 Introduction

2556 The functional interactions between a Client and a Server are described in 11.1 through 11.4
2557 respectively. The functional interactions use CRUDN messages (clause 8) and include Discovery,
2558 Notification, and Device management. These functions require support of core defined Resources
2559 as defined in Table 20.

2560

Table 20 – List of Core Resources

Pre-defined URI	Resource Name	Resource Type	Related Functional Interaction	Mandatory
"/oic/res"	Default	"oic.wk.res"	Discovery	Yes
"/oic/p"	Platform	"oic.wk.p"	Discovery	Yes
"/oic/d"	Device	"oic.wk.d"	Discovery	Yes
Implementation defined	Introspection	"oic.wk.introspection"	Introspection	Yes

2561

2562 **11.2 Resource discovery**

2563 **11.2.1 Introduction**

2564 Discovery is a function which enables OCF Endpoint discovery as well as Resource based
2565 discovery. OCF Endpoint discovery is described in detail in clause 10. This clause mainly describes
2566 the Resource based discovery.

2567 **11.2.2 Resource based discovery: mechanisms**

2568 **11.2.2.1 Overview**

2569 As part of discovery, a Client may find appropriate information about other OCF peers. This
2570 information could be instances of Resources, Resource Types or any other information represented
2571 in the Resource model that an OCF peer would want another OCF peer to discover.

2572 At the minimum, Resource based discovery uses the following:

- 2573 – A Resource to enable discovery shall be defined. The representation of that Resource shall
2574 contain the information that can be discovered.
- 2575 – The Resource to enable discovery shall be specified and commonly known a-priori. A Device
2576 for hosting the Resource to enable discovery shall be identified.
- 2577 – A mechanism and process to publish the information that needs to be discovered with the
2578 Resource to enable discovery.
- 2579 – A mechanism and process to access and obtain the information from the Resource to enable
2580 discovery. A query may be used in the request to limit the returned information.
- 2581 – A scope for the publication.
- 2582 – A scope for the access.
- 2583 – A policy for visibility of the information.

2584 Depending on the choice of the base aspects, the Framework defines three Resource based
2585 discovery mechanisms:

- 2586 – Direct discovery, where the Resources are published locally at the Device hosting the
2587 Resources and are discovered through peer inquiry.
- 2588 – Indirect discovery, where Resources are published at a third party assisting with the discovery
2589 and peers publish and perform discovery against the Resource to enable discovery on the
2590 assisting 3rd party.
- 2591 – Advertisement discovery, where the Resource to enable discovery is hosted local to the initiator
2592 of the discovery inquiry but remote to the Devices that are publishing discovery information.

2593 A Device shall support direct discovery.

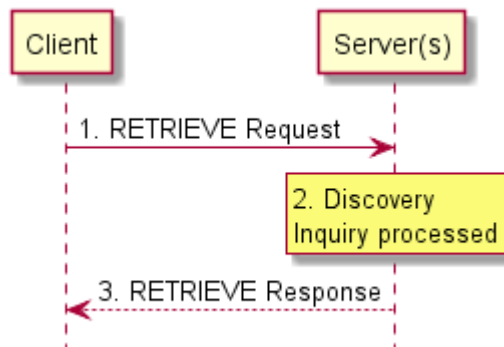
2594 **11.2.2.2 Direct discovery**

2595 In direct discovery,

- 2596 – The Device that is providing the information shall host the Resource to enable discovery.
- 2597 – The Device publishes the information available for discovery with the local Resource to enable
2598 discovery (i.e. local scope).
- 2599 – Clients interested in discovering information about this Device shall issue RETRIEVE requests
2600 directly to the Resource. The request may be made as a unicast or multicast. The request may
2601 be generic or may be qualified or limited by using appropriate queries in the request.
- 2602 – The Server Device that receives the request shall send a response with the discovered
2603 information directly back to the requesting Client Device.
- 2604 – The information that is included in the request is determined by the policies set for the Resource
2605 to be discovered locally on the responding Device.

2606 **11.2.3 Resource based discovery: Finding information**

2607 The discovery process (Figure 10) is initiated as a RETRIEVE request to the Resource to enable
2608 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as
2609 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the
2610 support in the data connectivity layer. The response to the request has the information to be
2611 discovered based on the policies for that information. The policies can determine which information
2612 is shared, when and to which requesting agent. The information that can be discovered can be
2613 Resources, types, configuration and many other standards or custom aspects depending on the
2614 request to appropriate Resource and the form of request. Optionally the requester may narrow the
2615 information to be returned in the request using query parameters in the URI query.



2616

2617 **Figure 10 – Resource based discovery: Finding information**

2618

2619 *Discovery Resources*

2620 The following Core Resources shall be implemented on all Devices to support discovery:

- 2621 – "/oic/res" for discovery of Resources.
- 2622 – "/oic/p" for discovery of Platform.
- 2623 – "/oic/d" for discovery of Device information.

2624 Devices shall expose each of "/oic/res", "/oic/d", and "/oic/p" via an unsecured OCF Endpoint.
2625 Further details for these mandatory Core Resources are described in Table 21.

2626 *Platform Resource*

2627 The OCF recognizes that more than one instance of Device may be hosted on a single Platform.
 2628 Clients need a way to discover and access the information on the Platform. The Core Resource,
 2629 "/oic/p" exposes Platform specific Properties. All instances of Device on the same Platform shall
 2630 have the same values of any Properties exposed (i.e. a Device may choose to expose optional
 2631 Properties within "/oic/p" but when exposed the value of that Property should be the same as the
 2632 value of that Property on all other Devices on that Platform).

2633 *Device Resource*

2634 The Device Resource shall have the pre-defined URI "/oic/d", the Device Resource shall expose
 2635 the Properties pertaining to a Device as defined in Table 24. The Device Resource shall have a
 2636 default Resource Type that helps in bootstrapping the interactions with the Device (the default type
 2637 is described in Table 21).The Device Resource may have one or more Resource Type(s) that are
 2638 specific to the Device in addition to the default Resource Type or if present overriding the default
 2639 Resource Type. The base Resource Type "oic.wk.d" defines the Properties that shall be exposed
 2640 by all Devices. The Device specific Resource Type(s) exposed are dependent on the class of
 2641 Device (e.g. air conditioner, smoke alarm, etc. Since all the Resource Types of "/oic/d" are not
 2642 known a priori, the Resource Type(s) of "/oic/d" are determined by discovery through the Core
 2643 Resource "/oic/res".

2644 **Table 21 – Mandatory discovery Core Resources**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
"/oic/res"	Default	"oic.wk.res"	"oic.if.ll"	The Resource through which the corresponding Server is discovered and introspected for available Resources. "/oic/res" shall expose the Resources that are discoverable on a Device. When a Server receives a RETRIEVE request targeting "/oic/res" (e.g., "GET /oic/res"), it shall respond with the links list of all the Discoverable Resources of itself. The "/oic/d" and "/oic/p" are Discoverable Resources, hence their links are included in "/oic/res" response. The Properties exposed by "/oic/res" are listed in Table 22.	Discovery
"/oic/p"	Platform	"oic.wk.p"	"oic.if.r"	The Discoverable Resource through which Platform specific information is discovered. The Properties exposed by "/oic/p" are listed in Table 25	Discovery
"/oic/d"	Device	"oic.wk.d" and/or one or more Device Specific Resource Type ID(s)	"oic.if.r"	The discoverable via "/oic/res" Resource which exposes Properties specific to the Device instance. The Properties exposed by "/oic/d" are listed in Table 24.	Discovery

2645 Table 22 defines "oic.wk.res" Resource Type.

2646 **Table 22 – "oic.wk.res" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	"n"	string	N/A	N/A	R	No	Human-friendly name defined by the vendor

Links	"links"	array	See 7.8.2	N/A	R	Yes	The array of Links describes the URI, supported Resource Types and OCF Interfaces, and access policy.
-------	---------	-------	-----------	-----	---	-----	---

2647

2648 A Device shall support CoAP based discovery as the baseline discovery mechanism (see 11.2.5).

2649 The "/oic/res" shall list all Resources that are indicated as discoverable (see 11.2). Also the
2650 following architecture Resource Types shall be listed:

- 2651 – Introspection Resource indicated with an "rt" value of "oic.wk.introspection".
- 2652 – "/oic/p" indicated with an "rt" value of "oic.wk.p".
- 2653 – "/oic/d" indicated with an "rt" value of "oic.wk.d"
- 2654 – "/oic/sec/doxm" indicated with an "rt" value of "oic.r.doxm" as defined in ISO/IEC 30118-2:2018.
- 2655 – "/oic/sec/pstat" indicated with an "rt" value of "oic.r.pstat" as defined in ISO/IEC 30118-2:2018.
- 2656 – "/oic/sec/acl2" indicated with an "rt" value of "oic.r.acl2" as defined in ISO/IEC 30118-2:2018.
- 2657 – "/oic/sec/cred" indicated with an "rt" value of "oic.r.cred" as defined in ISO/IEC 30118-2:2018.

2658 Conditionally required:

- 2659 – "/oic/res" with an "rt" value of "oic.wk.res" as self-reference, on the condition that "oic/res" has
2660 to signal that it is Observable by a Client.

2661 The Introspection Resource is only applicable for Devices that host Vertical Resource Types (e.g.
2662 "oic.r.switch.binary") or vendor-defined Resource Types. Devices that only host Resources
2663 required to onboard the Device as a Client do not have to implement the Introspection Resource.

2664 Table 23 provides an OCF registry for protocol schemes.

2665 **Table 23 – Protocol scheme registry**

SI Number	Protocol
1	"coap"
2	"coaps"
3	"http"
4	"https"
5	"coap+tcp"
6	"coaps+tcp"

2666

2667 NOTE The discovery of an OCF Endpoint used by a specific protocol is out of scope. The mechanism used by a Client
2668 to form requests in a different messaging protocol other than discovery is out of scope.

2669 The following applies to the use of "/oic/d":

- 2670 – A vertical may choose to extend the list of Properties defined by the Resource Type "oic.wk.d".
2671 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The
2672 mandatory Properties defined in Table 24 shall always be present.
- 2673 – A Device may choose to expose a separate, Discoverable Resource with its Resource Type ID
2674 set to a Device Type. In this case the Resource is equivalent to an instance of "oic.wk.d" and

2675 adheres to the definition thereof. As such the Resource shall at a minimum expose the
 2676 mandatory Properties of "oic.wk.d". In the case where the Resource tagged in this manner is
 2677 defined to be an instance of a Collection in accordance with 7.8.3 then the Resources that are
 2678 part of that Collection shall at a minimum include the Resource Types mandated for the Device
 2679 Type.

2680 Table 24 "oic.wk.d" Resource Type definition defines the base Resource Type for the "/oic/d"
 2681 Resource.

2682 **Table 24 – "oic.wk.d" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	"n"	"string"	N/A	N/A	R	Yes	Human friendly name defined by the vendor. In the presence of "n" Property of "/oic/con", both have the same Property Value. When "n" Property Value of "/oic/con" is modified, it shall be reflected to "n" Property Value of "/oic/d".
Spec Version	"icv"	"string"	N/A	N/A	R	Yes	The specification version of this document that a Device is implemented to. The syntax shall be "ocf.<major>.<minor>.<sub-version>" where <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of this document respectively. The specification version number (i.e., <major>.<minor>.<sub-version>) shall be obtained from the title page of this document (e.g. "2.0.5"). An example of the string value for this Property is "ocf.2.0.5".
Device ID	"di"	"uuid"	N/A	N/A	R	Yes	Unique identifier for Device. This value shall be the same value (i.e. mirror) as the doxm.deviceuuid Property as defined in ISO/IEC 30118-2:2018. Handling privacy-sensitivity for the "di" Property, refer to clause 13.16 in ISO/IEC 30118-2:2018.
Data Model Version	"dmv"	"csv"	N/A	N/A	R	Yes	Spec version of the Resource specification to which this Device data model is implemented; if implemented against a Vertical specific Device specification(s), then the Spec version of the vertical specification this Device model is implemented to. The syntax is a comma separated list of <res>.<major>.<minor>.<sub-version> or <vertical>.<major>.<minor>.<sub-version>. <res> is the string "ocf.res" and <vertical> is the name of the vertical defined in the Vertical specific Resource specification. The <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. One entry in the csv

							string shall be the applicable version of the Resource Type Specification for the Device (e.g. "ocf.res.1.0.0"). If applicable, additional entry(-ies) in the csv shall be the vertical(s) being realized (e.g. "ocf.sh.1.0.0"). This value may be extended by the vendor. The syntax for extending this value, as a comma separated entry, by the vendor shall be by adding x.<Domain_Name>.<vendor_string> . For example "ocf.res.1.0.0, ocf.sh.1.0.0, x.com.example.string", The order of the values in the comma separated string can be in any order (i.e. no prescribed order). This Property shall not exceed 256 octets.
Permanent Immutable ID	"piid"	"uuid"	N/A	N/A	R	Yes	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 Permanent Immutable ID value for all the protocols it supports. Handling privacy-sensitivity for the "piid" Property, refer to clause 13.16 in ISO/IEC 30118-2:2018.
Localized Descriptions	"ld"	"array"	N/A	N/A	R	No	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 IETF RFC 5646 language tag) and a "value" field containing the Device description in the indicated language.
Software Version	"sv"	"string"	N/A	N/A	R	No	Version of the Device software.
Manufacturer Name	"dmn"	"array"	N/A	N/A	R	No	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 IETF RFC 5646 language tag) and a "value" field containing the manufacturer name in the indicated language.
Model Number	"dmno"	"string"	N/A	N/A	R	No	Model number as designated by manufacturer.
Ecosystem Name	"econame"	"string"	enum	N/A	R	No	This is the name of ecosystem that a Bridged Device belongs to. If a Device has "oic.d.virtual" as one of Resource Type values ("rt") the Device shall contain this Property, otherwise this Property shall not be included. This Property has enumeration values: ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"].
Version of Ecosystem	"ecoversion"	"string"	N/A	N/A	R	No	This is the version of ecosystem that a Bridged Device belongs to. If a Device has "oic.d.virtual" as one of its Resource Type values ("rt")

							the Device should contain this Property, otherwise this Property shall not be included.
--	--	--	--	--	--	--	---

2683 Table 25 defines "oic.wk.p" Resource Type.

2684

Table 25 – "oic.wk.p" Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform ID	"pi"	"uuid"	N/A	N/A	R	Yes	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. Handling privacy-sensitivity for the "pi" Property, refer to clause 13.16 in ISO/IEC 30118-2:2018.
Manufacturer Name	"mnmn"	"string"	N/A	N/A	R	Yes	Name of manufacturer.
Manufacturer Details Link	"mnmli"	"uri"	N/A	N/A	R	No	Reference to manufacturer, represented as a URI.
Model Number	"mnmo"	"string"	N/A	N/A	R	No	Model number as designated by manufacturer.
Date of Manufacture	"mndt"	"date"	N/A	Time	R	No	Manufacturing date of Platform.
Serial number	"mnsel"	"string"	N/A	s	R	No	Serial number of the Platform, may be unique for each Platform of the same model number.
Platform Version	"mnpv"	"string"	N/A	N/A	R	No	Version of Platform – string (defined by manufacturer).
OS Version	"mnos"	"string"	N/A	N/A	R	No	Version of Platform resident OS – string (defined by manufacturer).
Hardware Version	"mnhw"	"string"	N/A	N/A	R	No	Version of Platform hardware.
Firmware version	"mfnv"	"string"	N/A	N/A	R	No	Version of Platform firmware.
Support link	"mnsli"	"uri"	N/A	N/A	R	No	URI that points to support information from manufacturer.
SystemTime	"st"	"date-time"	N/A	N/A	R	No	Reference time for the Platform.
Vendor ID	"vid"	"string"	N/A	N/A	R	No	Vendor defined string for the Platform. The string is freeform and up

							to the vendor on what text to populate it.
Network Connectivity Type	"mnnct"	"array"	array of integer		R	No	An array of integer where each integer indicates the network connectivity type based on IANAIfType value as defined by IANA ifType-MIB Definitions, e.g., [71, 259] which represents Wi-Fi and Zigbee.

2685 **11.2.4 Resource discovery using "/oic/res"**

2686 Discovery using "/oic/res" is the default discovery mechanism that shall be supported by all Devices
2687 as follows:

- 2688 – Every Device updates its local "/oic/res" with the Resources that are discoverable (see 7.3.2.2).
2689 Every time a new Resource is instantiated on the Device and if that Resource is discoverable
2690 by a remote Device then that Resource is published with the "/oic/res" Resource that is local to
2691 the Device (as the instantiated Resource).
- 2692 – A Device wanting to discover Resources or Resource Types on one or more remote Devices
2693 makes a RETRIEVE request to the "/oic/res" on the remote Devices. This request may be sent
2694 multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request may
2695 optionally be restricted using appropriate clauses in the query portion of the request. Queries
2696 may select based on Resource Types, OCF Interfaces, or Properties.
- 2697 – The query applies to the representation of the Resources. "/oic/res" is the only Resource whose
2698 representation has "rt". So "/oic/res" is the only Resource that can be used for Multicast
2699 discovery at the transport protocol layer.
- 2700 – The Device receiving the RETRIEVE request responds with a list of Resources, the Resource
2701 Type of each of the Resources and the OCF Interfaces that each Resource supports.
2702 Additionally, information on the policies active on the Resource can also be sent. The policy
2703 supported includes Observability and discoverability.
- 2704 – The receiving Device may do a deeper discovery based on the Resources returned in the
2705 request to "/oic/res".

2706 The information that is returned on discovery against "/oic/res" is at the minimum:

- 2707 – The URI (relative or fully qualified URL) of the Resource.
- 2708 – The Resource Type(s) of each Resource. More than one Resource Type may be returned if the
2709 Resource enables more than one type. To access Resources of multiple types, the specific
2710 Resource Type that is targeted shall be specified in the request.
- 2711 – The OCF Interfaces supported by that Resource. Multiple OCF Interfaces may be returned. To
2712 access a specific OCF Interface that OCF Interface shall be specified in the request. If the OCF
2713 Interface is not specified, then the Default OCF Interface is assumed.

2714 For Clients that do include the OCF-Accept-Content-Format-Version option, an "/oic/res" response
2715 includes an array of Links to conform to IETF RFC 6690. Each Link shall use an "eps" Parameter
2716 to provide the information for an encrypted connection and carry "anchor" of the value OCF URI
2717 where the authority component of <deviceID> indicates the Device hosting the target Resource.

2718 The OpenAPI 2.0 file for discovery using "/oic/res" is described in Annex A. Also refer to clause 10
2719 (OCF Endpoint discovery) for details of Multicast discovery using "/oic/res" on a CoAP transport.

2720 An example Device might return the following to Clients that request with the Content Format of
2721 "application/vnd.ocf+cbor" in Accept Option:

```

2722 [
2723   {
2724     "href": "/oic/res",
2725     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989/oic/res",
2726     "rel": "self",
2727     "rt": ["oic.wk.res"],
2728     "if": ["oic.if.ll", "oic.if.baseline"],
2729     "p": {"bm": 3},
2730     "eps": [{"ep": "coap://[fe80::b1d6]:44444"}]
2731   },
2732   {
2733     "href": "/oic/p",
2734     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2735     "rt": ["oic.wk.p"],
2736     "if": ["oic.if.r", "oic.if.baseline"],
2737     "p": {"bm": 3},
2738     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2739             {"ep": "coaps://[fe80::b1d6]:11111"}]
2740   },
2741 },
2742 {
2743   "href": "/oic/d",
2744   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2745   "rt": ["oic.wk.d"],
2746   "if": ["oic.if.r", "oic.if.baseline"],
2747   "p": {"bm": 3},
2748   "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2749           {"ep": "coaps://[fe80::b1d6]:11111"}]
2750 },
2751 },
2752 {
2753   "href": "/myLightSwitch",
2754   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2755   "rt": ["oic.r.switch.binary"],
2756   "if": ["oic.if.a", "oic.if.baseline"],
2757   "p": {"bm": 3},
2758   "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2759           {"ep": "coaps://[fe80::b1d6]:11111"}]
2760 },
2761 }
2762 ]

```

2763 After performing discovery using "/oic/res", Clients may discover additional details about Server by
2764 performing discovery using "/oic/p", etc. If a Client already knows about Server it may discover
2765 using other Resources without going through the discovery of "/oic/res".

2766 11.2.5 Multicast discovery using "/oic/res"

2767 Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices shall
2768 support use of CoAP multicast to allow retrieving the "/oic/res" Resource from an unsecured OCF
2769 Endpoint on the Device. Clients may support use of CoAP multicast to retrieve the "/oic/res"
2770 Resource from other Devices. The CoAP multicast retrieval of "/oic/res" supports filtering Links
2771 based on the "rt" Property in the Links:

- 2772 – If the discovery request is intended for a specific Resource Type including as part of a multi-
2773 value Resource Type, the query parameter "rt" shall be included in the request (see 6.2.2) with
2774 its value set to the desired Resource Type. Only Devices hosting the Resource Type shall
2775 respond to the discovery request.
- 2776 – When the "rt" query parameter is omitted, all Devices shall respond to the discovery request.

2777 **11.3 Notification**

2778 **11.3.1 Overview**

2779 A Server shall support NOTIFY operation to enable a Client to request and be notified of desired
2780 states of one or more Resources in an asynchronous manner. 11.3.2 specifies the Observe
2781 mechanism in which updates are delivered to the requester.

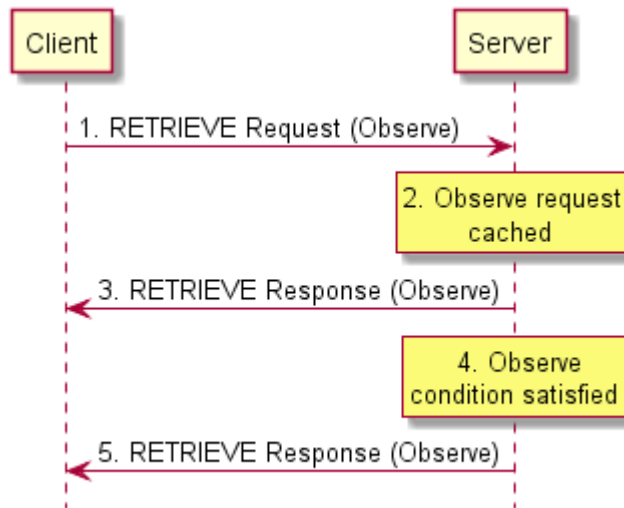
2782 **11.3.2 Observe**

2783 **11.3.2.1 Overview**

2784 In the Observe mechanism the Client utilizes the RETRIEVE operation to require the Server for
2785 updates in case of Resource state changes. The Observe mechanism consists of five steps which
2786 are depicted in Figure 11.

2787 NOTE the Observe mechanism can only be used for a resource with a Property of Observable (see 7.3.2.2).

2788



2789

2790

2791

Figure 11 – Observe Mechanism

2792 **11.3.2.2 RETRIEVE request with Observe indication**

2793 The Client transmits a RETRIEVE request message to the Server to request updates for the
2794 Resource on the Server if there is a state change. The RETRIEVE request message carries the
2795 following parameters:

- 2796 – *fr*: Unique identifier of the Client.
- 2797 – *to*: Resource that the Client is requesting to Observe.
- 2798 – *ri*: Identifier of the RETRIEVE operation.
- 2799 – *op*: RETRIEVE.
- 2800 – *obs*: Indication for Observe operation.

2801 **11.3.2.3 Processing by the Server**

2802 Following the receipt of the RETRIEVE request, the Server may validate if the Client has the
2803 appropriate rights for the requested operation and the Properties are readable and Observable. If
2804 the validation is successful, the Server caches the information related to the Observe request. The

2805 Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial
2806 response and future responses in case of a change of state.

2807 **11.3.2.4 RETRIEVE response with Observe indication**

2808 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
2809 message from a Client. If validation succeeded, the response includes an Observe indication. If
2810 not, the Observe indication is omitted from the response which signals to the requesting Client that
2811 registration for notification was not allowed.

2812 The RETRIEVE response message shall include the following parameters:

- 2813 – *fr*: Unique identifier of the Server.
- 2814 – *to*: Unique identifier of the Client.
- 2815 – *ri*: Identifier included in the RETRIEVE operation.
- 2816 – *cn*: Information Resource representation as requested by the Client.
- 2817 – *rs*: The result of the RETRIEVE operation.
- 2818 – *obs*: Indication that the response is made to an Observe operation.

2819 **11.3.2.5 Resource monitoring by the Server**

2820 The Server shall monitor the state the Resource identified in the Observe request from the Client.
2821 Anytime there is a change in the state of the Observed Resource, the Server sends another
2822 RETRIEVE response with the Observe indication. The mechanism does not allow the client to
2823 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

2824 **11.3.2.6 Additional RETRIEVE responses with Observe indication**

2825 The Server shall transmit updated RETRIEVE response messages following Observed changes in
2826 the state of the Resources indicated by the Client. The RETRIEVE response message shall include
2827 the parameters listed in 11.3.2.4.

2828 **11.3.2.7 Cancelling Observe**

2829 The Client can explicitly cancel Observe by sending a RETRIEVE request without the Observe
2830 indication field to the same Resource on the Server which it was Observing. For certain protocol
2831 mappings, the Client may also be able to cancel an Observe by ceasing to respond to the
2832 RETRIEVE responses.

2833 **11.4 Introspection**

2834 **11.4.1 Overview**

2835 Introspection is a mechanism to announce the capabilities of Resources hosted on the Device.

2836 The intended usage of the Introspection Device Data (IDD) is to enable dynamic Clients e.g. Clients
2837 that can use the IDD) to generate dynamically a UI or dynamically create translations of the hosted
2838 Resources to another eco-system. Other usages of Introspection is that the information can be
2839 used to generate Client code. The IDD is designed to augment the existing data already on the
2840 wire. This means that existing mechanisms need to be used to get a full overview of what is
2841 implemented in the Device. For example, the IDD does not convey information about Observability,
2842 since that is already conveyed with the "p" Property on the Links in "/oic/res" (see 7.8.2.5.3).

2843 The IDD is recommended to be conveyed as static data. Meaning that the data does not change
2844 during the uptime of a Device. However, when the IDD is not static, the Introspection Resource
2845 shall be Observable and the url Property Value of "oic.wk.introspection" Resource shall change to
2846 indicate that the IDD is changed.

2847 The IDD describes the Resources that make up the Device. For the complete list of included
2848 Resources see Table 20. The IDD is described as a OpenAPI 2.0 in JSON format file. The text in
2849 the following bulleted list contains OpenAPI 2.0 terms, such as paths, methods etc. The OpenAPI
2850 2.0 file shall contain the description of the Resources:

- 2851 – The IDD will use the HTTP syntax, e.g., define the CRUDN operation as HTTP methods and
2852 use the HTTP status codes.
- 2853 – The IDD does not have to define all the status codes that indicate an error situation.
- 2854 – The IDD does not have to define a schema when the status code indicates that there is no
2855 payload (see HTTP status code 204 as an example).
- 2856 – The paths (URLs) of the Resources in the IDD shall be without the OCF Endpoint description,
2857 e.g. it shall not be a fully-qualified URL but only the relative path from the OCF Endpoint, aka
2858 the "href". The relative path may include a query parameter (e.g. "?if=oic.if.ll"), in such cases
2859 the text following (and including) the "?" delimiter shall be removed before equating to the "href"
2860 that is conveyed by "/oic/res".
- 2861 – The following Resources shall be excluded in the IDD:
 - 2862 – Resource with Resource Type: "oic.wk.res" unless 3rd party defined or optional Properties
2863 are implemented.
 - 2864 – Resource with Resource Type: "oic.wk.introspection".
 - 2865 – Resources explicitly identified within other specifications working in conjunction with this
2866 document (e.g. Resources that handle Wi-Fi Easy Setup, see [2]).
- 2867 – The following Resources shall be included in the IDD when optional or 3rd party defined
2868 Properties are implemented:
 - 2869 – Resources with type: "oic.wk.p" and "oic.wk.d" (e.g. discovery related Resources).
 - 2870 – Security Virtual Resources from ISO/IEC 30118-2:2018.
- 2871 – When the Device does not expose instances of Vertical Resource Types, and does not have
2872 any 3rd party defined Resources (see 7.8.4.4), and does not need to include Resources in the
2873 IDD due to other clauses in this clause, then the IDD shall be an empty OpenAPI 2.0 file. An
2874 example of an empty OpenAPI 2.0 file can be found in found in Annex **B.2**.
- 2875 – All other Resources that are individually addressable by a Client (i.e. the "href" can be resolved
2876 and at least one operation is supported with a success path response) shall be listed in the IDD.
- 2877 – Per Resource the IDD shall include:
 - 2878 – All implemented methods
 - 2879 – For an OCF defined Resource Type, only the methods that are listed in the OpenAPI 2.0
2880 definition are allowed to exist in the IDD. For an OCF defined Resource Type, methods
2881 not listed in the OpenAPI 2.0 definition shall not exist in the IDD. The supported methods
2882 contained in the IDD shall comply with the listed OCF Interfaces. For example, if the
2883 POST method is listed in the IDD, then an OCF Interface that allows UPDATE will be
2884 listed in the IDD.
 - 2885 – Per supported method:
 - 2886 – Implemented query parameters per method.
 - 2887 – This includes the supported OCF Interfaces ("if") as enum values.
 - 2888 – Schemas of the payload for the request and response bodies of the method.
 - 2889 – Where the schema provides the representation of a batch request or response ("oic.if.b")
2890 the schema shall contain the representations for all Resource Types that may be
2891 included within the batch representation. The representations shall be provided within
2892 the IDD itself.

- 2893 – The schema data shall be conveyed by the OpenAPI 2.0 schema.
- 2894 – The OpenAPI 2.0 schema object shall comply with:
 - 2895 – The schemas shall be fully resolved, e.g. no references shall exist outside the
 - 2896 OpenAPI 2.0 file.
 - 2897 – The schemas shall list which OCF Interfaces are supported on the method.
 - 2898 – The schemas shall list if a Property is optional or required.
 - 2899 – The schemas shall include all Property validation keywords. Where an enum is
 - 2900 defined the enum shall contain the values supported by the Device. When vendor
 - 2901 defined extensions exist to the enum (defined in accordance to 7.8.4.4) these shall
 - 2902 be included in the enum.
 - 2903 – The schemas shall indicate if an Property is read only or read-write.
 - 2904 – By means of the readOnly schema tag belonging to the Property.
 - 2905 – Default value of readOnly is false as defined by OpenAPI 2.0.
 - 2906 – The default value of the "rt" Property shall be used to indicate the supported
 - 2907 Resource Types.
 - 2908 – oneOf and anyOf constructs are allowed to be used as part of a OpenAPI 2.0 schema
 - 2909 object. The OpenAPI 2.0 schema with oneOf and anyOf constructs can be found in
 - 2910 Annex **B.1**.
- 2911 – For Atomic Measurements (see clause 7.8.4), the following apply:
 - 2912 – The "rts" Property Value in the IDD shall include only the Resource Types the instance
 - 2913 contains and not the theoretical maximal set allowed by the schema definition.
 - 2914 – The Resources that are part of an Atomic Measurement, excluding the Atomic Measurement
 - 2915 Resource itself, shall not be added to their own individual path in the IDD, as they are not
 - 2916 individually addressable; however, the schemas for the composed Resource Types shall be
 - 2917 provided in the IDD as part of the batch response definition along with the "href" for the
 - 2918 Resource.

2919 Dynamic Resources (e.g. Resources that can be created on a request by a Client) shall have a
 2920 URL definition which contains a URL identifier (e.g. using the {} syntax). A URL with {} identifies
 2921 that the Resource definition applies to the whole group of Resources that may be created. The
 2922 actual path may contain the Collection node that links to the Resource.

2923 Example of a URL with identifiers:

2924 /SceneListResURI/{SceneCollectionResURI}/{SceneMemberResURI}:

2925 When different Resource Types are allowed to be created in a Collection, then the different
 2926 schemas for the CREATE method shall define all possible Resource Types that may be created.
 2927 The schema construct oneOf allows the definition of a schema with selectable Resources. The
 2928 oneOf construct allows the integration of all schemas and that only one existing sub schema shall
 2929 be used to indicate the definition of the Resource that may be created.

2930 Example usage of oneOf JSON schema construct is shown in Figure 12:

```

2931 {
2932   "oneOf": [
2933     { <<subschema 1 definition>> },
2934     { << sub schema 2 definition >> }
2935     ...
2936   ]
2937 }
  
```

2938 **Figure 12 – Example usage of oneOf JSON schema**

2939 A Client using the IDD of a Device should check the version of the supported IDD of the Device.
 2940 The OpenAPI 2.0 version is indicated in each file with the tag "swagger". Example of the 2.0
 2941 supported version of the tag is: "swagger": "2.0". Later versions of this document may reference
 2942 newer versions of the OpenAPI specification, for example 3.0.

2943 A Device shall support one Resource with a Resource Type of "oic.wk.introspection" as defined in
 2944 Table 26. The Resource with a Resource Type of "oic.wk.introspection" shall be included in the
 2945 Resource "/oic/res".

2946 An empty IDD file, e.g. no URLs are exposed, shall still have the mandatory OpenAPI 2.0 fields.
 2947 See OpenAPI specification. An example of an empty OpenAPI 2.0 file can be found in found in
 2948 Annex B.2.

2949 **Table 26 – Introspection Resource**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
none	Introspection	"oic.wk.introspection"	"oic.if.r"	The Resource that announces the URL of the Introspection file.	Introspection

2950

2951 Table 27 defines "oic.wk.introspection" Resource Type.

2952 **Table 27 – "oic.wk.introspection" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
urlInfo	"urlInfo"	"array"	N/A	N/A	R	Yes	array of objects
url	"url"	"string"	"uri"	N/A	R	Yes	URL to the hosted payload
protocol	"protocol"	"string"	"enum"	N/A	R	Yes	Protocol definition to retrieve the Introspection Device Data from the url.
content-type	"content-type"	"string"	"enum"	N/A	R	No	content type of the url.
version	"version"	"integer"	"enum"	N/A	R	No	Version of the Introspection protocol, indicates which rules are applied on the Introspection Device Data regarding the content of the OpenAPI 2.0 file. Current value is 1.

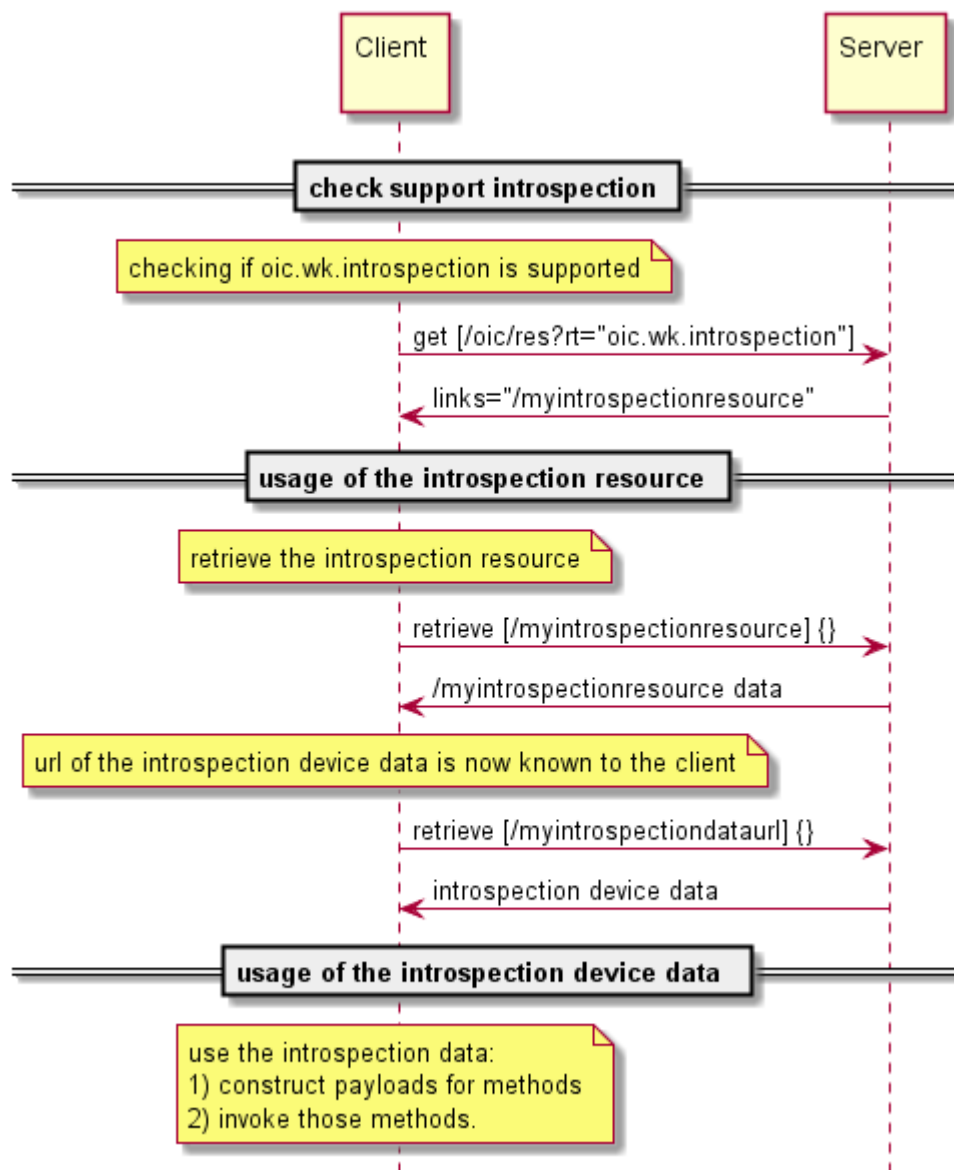
2953

2954 **11.4.2 Usage of Introspection**

2955 The Introspection Device Data is retrieved in the following steps and as depicted in Figure 13:

- 2956 – Check if the Introspection Resource is supported and retrieve the URL of the Resource.
- 2957 – Retrieve the contents of the Introspection Resource
- 2958 – Download the Introspection Device Data from the URL specified the Introspection Resource.
- 2959 – Usage of the Introspection Device Data by the Client

2960



2961

Figure 13 – Interactions to check Introspection support and download the Introspection Device Data.

2962
2963

2964 11.5 Semantic Tags

2965 11.5.1 Introduction

2966 Semantic Tags are meta-information associated with a specific Resource instance that are
2967 represented as both Link Parameters and Resource Properties that provide a mechanism whereby
2968 the Resource be annotated with additional contextual metadata that helps describe the Resource.

2969 When a Semantic Tag is defined for a Resource, it shall be present as a Link Parameter in all Links
2970 that are present that target the Resource, including Links in "/oic/res" if the Resource is a
2971 Discoverable Resource. The Semantic Tag is further treated as a Common Property associated
2972 with the Resource and so shall be returned as part of the "baseline" response for the Resource if
2973 a Semantic Tag has been populated.

2974 **11.5.2 Semantic Tag definitions**

2975 **11.5.2.1 Relative and descriptive position Semantic Tags**

2976 **11.5.2.1.1 Introduction**

2977 Consider where there may be multiple instances of the same Resource Type exposed by a Device;
2978 or a case where there may be potentially ambiguity with regard to the physical attribute that a
2979 Resource is representing. In such a case the ability to annotate the Links to the Resource with
2980 information pertaining to the relative position of the Resource within the Physical Device becomes
2981 useful.

2982 **11.5.2.1.2 "tag-pos-desc" or position description Semantic Tag**

2983 The "tag-pos-desc" Semantic Tag as defined in Table 28 describes the position of the Resource as
2984 a descriptive position. If the tag is not exposed it conveys the same meaning as if the tag is exposed
2985 with a value of "unknown". The value for the "tag-pos-desc" Semantic Tag if exposed, shall be a
2986 string containing a value from the enumeration detailed in Annex C. The population of the Semantic
2987 Tag is defined by the Device vendor and shall not be mutable by a Client.

2988 **Table 28 – "tag-pos-desc" Semantic Tag definition**

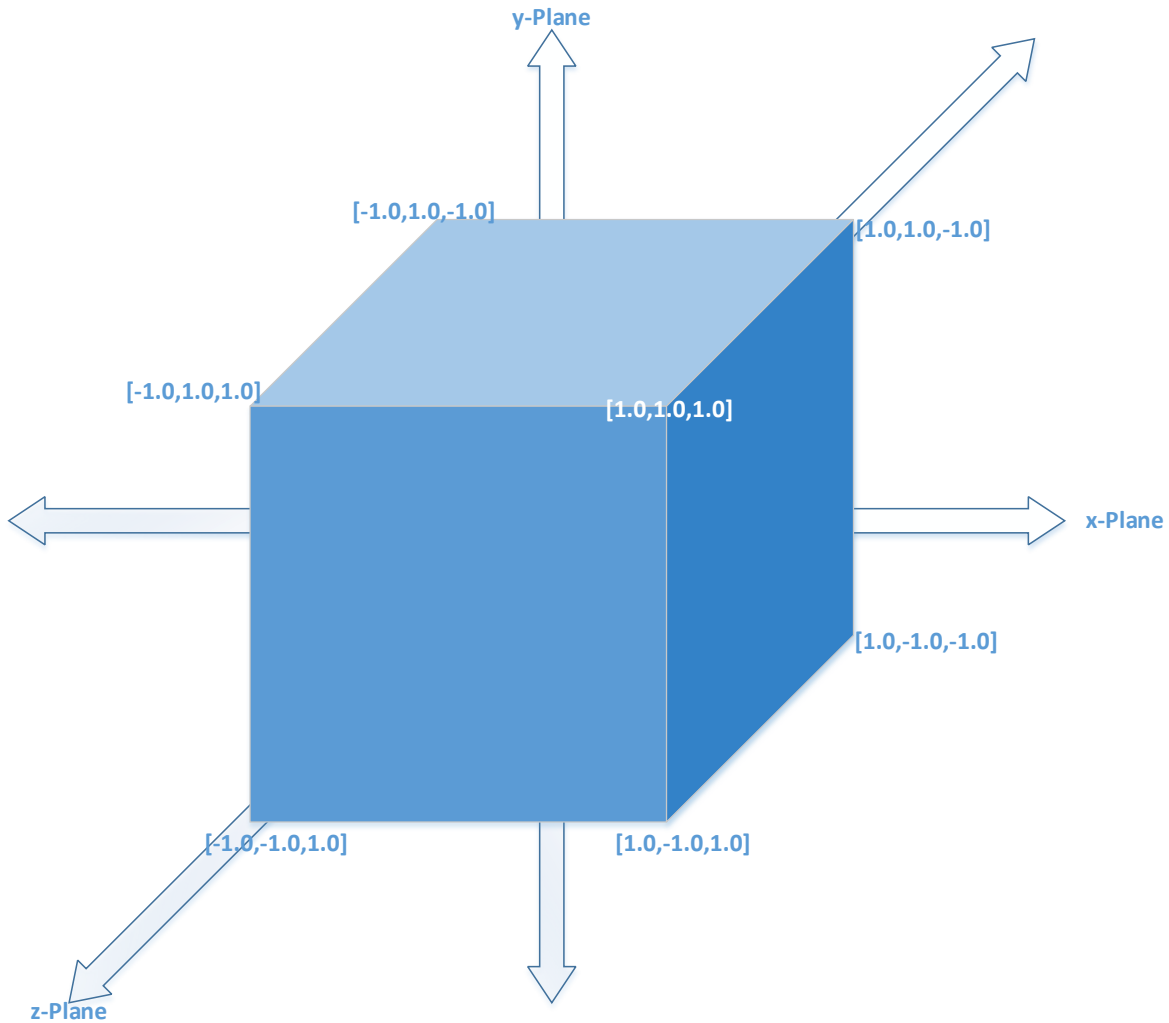
Link Parameter name	Type	Contents	Value example
"tag-pos-desc"	enum	See Annex C	"tag-pos-desc": "topleft"

2989

2990 **11.5.2.1.3 "tag-pos-rel" or relative position Semantic Tag**

2991 The "tag-pos-rel" Semantic Tag describes the position of the Resource as a relative position in 3D
2992 space against a known point defined by the Device vendor. The known point is defined using [x,y,z]
2993 form as [0.0,0.0,0.0]. The position itself is then represented by the x-, y-, and z- plane relative
2994 position from this known point using a bounded box of size +1.0/-1.0 in each plane.

2995 Figure 14 illustrates the definition of "tag-pos-rel".



2996

2997

Figure 14 – "tag-pos-rel" definition

2998 The "tag-pos-rel" Semantic Tag value is defined by the Device vendor and shall not be mutable by
 2999 a Client. This is detailed in Table 29.

3000

Table 29 – "tag-pos-rel" Semantic Tag definition

Link Parameter name	Type	Contents	Value example
"tag-pos-rel"	array	Three element array of numbers defining the position relative to a known [0,0,0] point within the context of an abstract box [-1,-1,-1],[1,1,1].	"tag-pos-rel": [0.5,0.5,0.5]

3001

3002 11.5.2.2 Functional behaviour Semantic Tags

3003 11.5.2.2.1 Introduction

3004 Consider, for example, the case of a Device that supports two target temperatures simultaneously for
 3005 different modes of operation, for example a temperature for heating and a separate temperature
 3006 for cooling.

3007 There is then an ambiguity with respect to the target mode of the specific temperature Resource;
3008 it isn't explicit which instance of temperature is associated with which Device function. In such a
3009 case the ability to annotate the Links to the Resource with information pertaining to the function of
3010 the Resource within the Physical Device becomes useful.

3011 **11.5.2.2.2 "tag-func-desc" or function description Semantic Tag**

3012 The "tag-func-desc" Semantic Tag describes the function of the Resource, if exposed it shall be
3013 populated with a value from the currently supported set of standardized enumeration values defined
3014 by the Device ecosystem specifications. If the tag is not exposed it conveys the same meaning as
3015 if the tag is exposed with a value of "unknown". The value for the "tag-func-desc" Semantic Tag, if
3016 exposed, is defined by the Device vendor and shall not be mutable by a Client.

3017 This "tag-func-desc" Semantic Tag is detailed in Table 30.

3018 **Table 30 – "tag-func-desc" Semantic Tag definition**

Link Parameter name	Type	Contents	Value example
"tag-func-rel"	enum	Defined by Device ecosystem	"tag-func-desc": "cool"

3019

3020 **12 Messaging**

3021 **12.1 Introduction**

3022 This clause specifies the protocol messaging mapping to the CRUDN messaging operations (clause
3023 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols is expected
3024 in later version of this document. All the Property information from the Resource model shall be
3025 carried within the message payload. This payload shall be generated in the Resource model layer
3026 and shall be encapsulated in the data connectivity layer. The message header shall only be used
3027 to describe the message payload (e.g., verb, mime-type, message payload format), in addition to
3028 the mandatory header fields defined in a messaging protocol (e.g., CoAP) specification. If the
3029 message header does not support this, then this information shall also be carried in the message
3030 payload. Resource model information shall not be included in the message header structure unless
3031 the message header field is mandatory in the messaging protocol specification.

3032 When a Resource is specified with a RESTful description language like OpenAPI 2.0 then the HTTP
3033 syntax definitions are used in the description (e.g., HTTP syntax for the CRUDN operations, status
3034 codes, etc). The HTTP syntax will be mapped to the actual used web transfer protocol (e.g., CoAP).

3035 **12.2 Mapping of CRUDN to CoAP**

3036 **12.2.1 Overview**

3037 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in clause
3038 12.2.3. A Device implementing CoAP shall conform to IETF RFC 7641 to implement the CoAP
3039 Observe option. Support for CoAP block transfer when the payload is larger than the MTU is defined
3040 in 12.2.8.

3041 **12.2.2 URIs**

3042 An OCF: URI is mapped to a coap: URI by replacing the scheme name "ocf" with "coap" if unsecure
3043 or "coaps" if secure before sending over the network by the requestor. Similarly on the receiver
3044 side, the scheme name is replaced with "ocf".

3045 Any query string that is present within the URI is encoded as one or more URI-Query Options as
3046 defined in IETF RFC 7252 clause 6.4.

3047 **12.2.3 CoAP method with request and response**

3048 **12.2.3.1 Overview**

3049 Every request has a CoAP method that realizes the request. The primary methods and their
3050 meanings are shown in Table 31, which provides the mapping of GET/POST/DELETE methods to
3051 CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides the generic
3052 behaviours when using these methods, however Resource OCF Interfaces may modify these
3053 generic semantics. The HTTP codes in the RESTful descriptions will be translated as described in
3054 IETF RFC 8075 clause 7 Response Code Mapping. CoAP methods not listed in Table 31 are not
3055 supported.

3056 **Table 31 – CoAP request and response**

Method for CRUDN	(mandatory) Request data	(mandatory) Response data
GET for RETRIEVE	- Method code: GET (0.01). - Request URI: an existing URI for the Resource to be retrieved	- Response code: success (2.xx) or error (4.xx or 5.xx). - Payload: Resource representation of the target Resource (when successful).
POST for CREATE	- Method code: POST (0.02). - Request URI: an existing URI for the Resource responsible for the creation. - Payload: Resource presentation of the Resource to be created.	- Response code: success (2.xx) or error (4.xx or 5.xx). - Payload: the URI of the newly created Resource (when successful).
POST for UPDATE	- Method code: POST (0.02). - Request URI: an existing URI for the Resource to be updated. - Payload: representation of the Resource to be updated.	- Response Code: success (2.xx) or error (4.xx or 5.xx).
DELETE for DELETE	- Method code: DELETE (0.04). - Request URI: an existing URI for the Resource to be deleted.	- Response code: success (2.xx) or error (4.xx or 5.xx).

3057

3058

3059 **12.2.3.2 CREATE with POST**

3060 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
3061 existing Resource on the Server that is processing the request. If no such Resource is present, the
3062 Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall use
3063 an existing request URI which identifies the Resource on the Server responsible for creation. The
3064 URI of the created Resource is determined by the Server and provided to the Client in the response.

3065 A Client shall include the representation of the new Resource in the request payload. The new
3066 resource representation in the payload shall have all the necessary Properties to create a valid
3067 Resource instance, i.e. the created Resource should be able to properly respond to the valid
3068 Request with mandatory OCF Interface (e.g., "GET with ?if=oc.if.baseline").

3069 Upon receiving the POST request, the Server shall either:

- 3070 – Create the new Resource with a new URI, respond with the new URI for the newly created
3071 Resource and a success response code (2.xx); or
- 3072 – respond with an error response code (4.xx or 5.xx).

3073 **12.2.3.3 RETRIEVE with GET**

3074 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of
3075 the target Resource identified by the request URI.

3076 Upon receiving the GET request, the Server shall either:

- 3077 – Send back the response with the representation of the target Resource with a success response
3078 code (2.xx); or
- 3079 – respond with an error response code (4.xx or 5.xx) or ignore it (e.g. non-applicable multicast
3080 GET).

3081 GET is a safe method and is idempotent.

3082 **12.2.3.4 UPDATE with POST**

3083 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
3084 existing Resource on the Server that is processing the request. If no such Resource is present, the
3085 Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE
3086 Property values of an existing Resource.

3087 Upon receiving the request, the Server shall either:

- 3088 – Apply the request to the Resource identified by the request URI in accordance with the applied
3089 OCF Interface (i.e. POST for non-existent Properties is ignored) and send back a response with
3090 a success response code (2.xx); or
- 3091 – respond with an error response code (4.xx or 5.xx). Note that if the representation in the payload
3092 is incompatible with the target Resource for POST using the applied OCF Interface (i.e. the
3093 overwrite semantic cannot be honored because of read-only Property in the payload), then the
3094 error response code 4.xx shall be returned.

3095 **12.2.3.5 DELETE with DELETE**

3096 DELETE shall be used for DELETE operation. The DELETE method requests that the Resource
3097 identified by the request URI be deleted.

3098 Upon receiving the DELETE request, the Server shall either:

- 3099 – Delete the target Resource and send back a response with a success response code (2.xx); or
- 3100 – respond with an error response code (4.xx or 5.xx).

3101 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

3102 **12.2.4 Content-Format negotiation**

3103 The Framework mandates support of CBOR, however it allows for negotiation of the payload body
3104 if more than one Content-Format (e.g. CBOR and JSON) is supported by an implementation. In this
3105 case the Accept Option defined in clause 5.10.4 of IETF RFC 7252 shall be used to indicate which
3106 Content-Format (e.g. JSON) is requested by the Client.

3107 The Content-Formats supported are shown in Table 32.

3108 **Table 32 – OCF Content-Formats**

Media Type	ID
"application/vnd.ocf+cbor"	10000

3109

3110 Clients shall include a Content-Format Option in every message that contains a payload. Servers
 3111 shall include a Content-Format Option for all success (2.xx) responses with a payload body. Per
 3112 IETF RFC 7252 clause 5.5.1, Servers shall include a Content-Format Option for all error (4.xx or
 3113 5.xx) responses with a payload body unless they include a Diagnostic Payload; error responses
 3114 with a Diagnostic Payload do not include a Content-Format Option. The Content-Format Option
 3115 shall use the ID column numeric value from Table 32. An OCF vertical may mandate a specific
 3116 Content-Format Option.

3117 Clients shall also include an Accept Option in every request message. The Accept Option shall
 3118 indicate the required Content-Format as defined in Table 32 for response messages. The Server
 3119 shall return the required Content-Format if available. If the required Content-Format cannot be
 3120 returned, then the Server shall respond with an appropriate error message.

3121 **12.2.5 OCF-Content-Format-Version information**

3122 Servers and Clients shall include the OCF-Content-Format-Version Option in both request and
 3123 response messages with a payload. Clients shall include the OCF-Accept-Content-Format-Version
 3124 Option in request messages. The OCF-Content-Format-Version Option and OCF-Accept-Content-
 3125 Format-Version Option are specified as Option Numbers in the CoAP header as shown in Table 33.

3126 **Table 33 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option**
 3127 **Numbers**

CoAP Option Number	Name	Format	Length (bytes)
2049	OCF-Accept-Content-Format-Version	uint	2
2053	OCF-Content-Format-Version	uint	2

3128
 3129 The value of both the OCF-Accept-Content-Format-Version Option and the OCF-Content-Format-
 3130 Version Option is a two-byte unsigned integer that is used to define the major, minor and sub
 3131 versions. The major and minor versions are represented by 5 bits and the sub version is
 3132 represented by 6 bits as shown in Table 34.

3133 **Table 34 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version**
 3134 **Representation**

	Major Version					Minor Version					Sub Version					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

3135
 3136 Table 35 illustrates several examples:

3137 **Table 35 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-**
 3138 **Version Representation**

OCF version	Binary representation	Integer value
"1.0.0"	"0000 1000 0000 0000"	2048
"1.1.0"	"0000 1000 0100 0000"	2112

3140 The OCF-Accept-Content-Format-Version Option and OCF-Content-Format-Version Option for this
3141 version of the document shall be "1.0.0" (i.e. "0b0000 1000 0000 0000").

3142 12.2.6 Content-Format policy

3143 All Devices shall support the current Content-Format Option, "application/vnd.ocf+cbor", and OCF-
3144 Content-Format-Version "1.0.0".

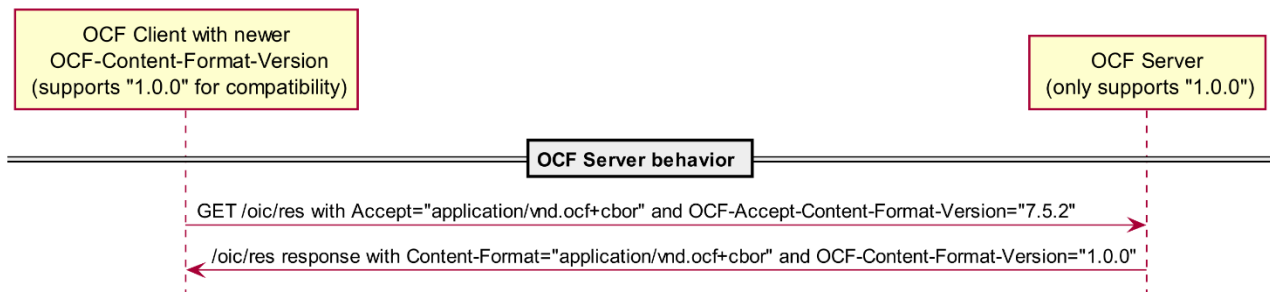
3145 For backward compatibility with previous OCF-Content-Format-Version Options:

- 3146 – All Client Devices shall support OCF-Content-Format-Version Option set to "1.0.0" and higher.
- 3147 – All Client Devices shall support OCF-Accept-Content-Format-Version Option set to "1.0.0" and
3148 higher.
- 3149 – A Client shall send a discovery request message with its Accept Option set to
3150 "application/vnd.ocf+cbor", and its OCF-Accept-Content-Format-Version Option matching its
3151 highest supported version.
- 3152 – A Server shall respond to a Client's discovery request that is higher than its OCF-Content-
3153 Format-Version by responding with its Content-Format Option set to "application/vnd.ocf+cbor",
3154 and OCF-Content-Format-Version matching its highest supported version. The response
3155 representation shall be encoded with the OCF-Content-Format-Version matching the Server's
3156 highest supported version.
- 3157 – A Server may support previous Content-Formats and OCF-Content-Format-Versions to support
3158 backward compatibility with previous versions.
- 3159 – For a Server that supports multiple OCF-Content-Format-Version Options, the Server should
3160 attempt to respond with an OCF-Content-Format-Version that matches the OCF-Accept-
3161 Content-Format-Version of the request.

3162 To maintain compatibility between Devices implemented to different versions of this document,
3163 Devices should follow the policy as described in Figure 15.

3164 The OCF Clients in Figure 15 support sending Content-Format Option set to
3165 "application/vnd.ocf+cbor", Accept Option set to "application/vnd.ocf+cbor", OCF-Content-Format-
3166 Version Option set to "1.0.0", and OCF-Accept-Content-Format-Version Option set to "1.0.0"
3167 (representing OCF 1.0 and later Clients). The OCF Servers in Figure 15 support sending Content-
3168 Format Option set to "application/vnd.ocf+cbor" and OCF-Content-Format-Version Option set to
3169 "1.0.0" (representing OCF 1.0 and later Servers).

3170



3171
3172 **Figure 15 – Content-Format Policy for backward compatible OCF Clients negotiating lower**
3173 **OCF Content-Format-Version**

3174 12.2.7 CRUDN to CoAP response codes

3175 The mapping of CRUDN operations response codes to CoAP response codes are identical to the
3176 response codes defined in IETF RFC 7252.

3177 **12.2.8 CoAP block transfer**

3178 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT
3179 devices. However scenarios can be envisioned in which an application needs to transfer larger
3180 payloads.

3181 CoAP block-wise transfer as defined in IETF RFC 7959 shall be used by all Servers which generate
3182 a content payload that would exceed the size of a CoAP datagram as the result of handling any
3183 defined CRUDN operation.

3184 Similarly, CoAP block-wise transfer as defined in IETF RFC 7959 shall be supported by all Clients.
3185 The use of block-wise transfer is applied to both the reception of payloads as well as transmission
3186 of payloads that would exceed the size of a CoAP datagram.

3187 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the
3188 same reliability setting (i.e. all confirmable or all non-confirmable).

3189 A Client may support both the block1 (as descriptive) and block2 (as control) options as described
3190 by IETF RFC 7959. A Server may support both the block1 (as control) and block2 (as descriptive)
3191 options as described by IETF RFC 7959.

3192 **12.2.9 Generic requirements for CoAP multicast**

3193 A Client may use CoAP multicast to retrieve a target Resource with a fixed local path from multiple
3194 other Devices. This clause provides generic requirements for this mechanism.

3195 – Devices shall join the All OCF Nodes multicast groups (as defined in [IANA IPv6 Multicast
3196 Address Space Registry]) with scopes 2, 3, and 5 (i.e., ff02::158, ff03::158 and ff05::158) and
3197 shall listen on the port 5683. For compliance to IETF RFC 7252 a Device may additionally join
3198 the All CoAP Nodes multicast groups.

3199 – Clients intending to discover Resources shall join the multicast groups as defined in the first
3200 bullet.

3201 – Clients shall send multicast requests to the All OCF Nodes multicast group address with scope
3202 2 ("ff02::158") at port "5683". The requested URI shall be the fixed local path of the target
3203 Resource optionally followed by query parameters. For compliance to IETF RFC 7252 a Client
3204 may additionally send to the All CoAP Nodes multicast groups.

3205 – To discover Devices on a low-rate wireless personal area network (LR-WPAN) [see
3206 IETF RFC 7346], Clients should send additional discovery requests (GET request) to the All
3207 OCF Nodes multicast group address with REALM_LOCAL scope 3 ("ff03::158") at port "5683".
3208 The set of replying Devices then can be used to distinguish if the Device is SITE_LOCAL or
3209 REALM_LOCAL to the Client discovering the Devices. Such request shall use the IPv6 hop limit
3210 with a value of 255. If the Client sends discovery requests to All OCF Nodes, then for
3211 compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast
3212 groups with the same REALM_LOCAL scope with the IPv6 hop limit value of 255.

3213 – Clients should send discovery requests (GET request) to the All OCF Nodes multicast group
3214 address with SITE_LOCAL scope 5 ("ff05::158") at port "5683". Such request shall use the IPv6
3215 hop limit with a value of 255. If the Client sends discovery requests to All OCF Nodes, then for
3216 compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast
3217 groups with the same SITE_LOCAL scope with the IPv6 hop limit value of 255.

3218 – The multicast request shall be permitted by matching the request to an ACE which permits
3219 unauthenticated access to the target Resource as described in ISO/IEC 30118-2:2018.

3220 – Handling of multicast requests shall be as described in clause 8 of IETF RFC 7252 and clause
3221 4.1 in IETF RFC 6690.

3222 – Devices which receive the request shall respond, subject to query parameter processing
3223 specific to the requested Resource.

3224 **12.3 Mapping of CRUDN to CoAP serialization over TCP**

3225 **12.3.1 Overview**

3226 In environments where TCP is already available, CoAP can take advantage of it to provide reliability.
3227 Also in some environments UDP traffic is blocked, so deployments may use TCP. For example,
3228 consider a cloud application acting as a Client and the Server is located at the user's home. A
3229 Server which already support CoAP as a messaging protocol could easily support CoAP
3230 serialization over TCP rather than utilizing another messaging protocol. A Device implementing
3231 CoAP Serialization over TCP shall conform to IETF RFC 8323.

3232 **12.3.2 URIs**

3233 When UDP is blocked, Clients are dependent on pre-configured details of the Device to determine
3234 if the Device supports CoAP serialization over TCP. When UDP is not-blocked, a Device which
3235 supports CoAP serialization over TCP shall populate the "eps" Parameter in the "/oic/res" response,
3236 as defined in 10.2, with the URI scheme(s) as defined in clause 8.1 or 8.2 of IETF RFC 8323. For
3237 the "coaps+tcp" URI scheme, as defined in clause 8.2 of IETF RFC 8323, IETF RFC 7301 shall be
3238 used. In addition, the URIs used for CoAP serialization over TCP shall conform to 12.2.2 by
3239 substituting the scheme names with the scheme names defined in clauses 8.1 and 8.2 of
3240 IETF RFC 8323 respectively.

3241 **12.3.3 CoAP method with request and response**

3242 The CoAP methods used for CoAP serialization over TCP shall conform to 12.2.3.

3243 **12.3.4 Content-Format negotiation**

3244 The Content Format negotiation used for CoAP serialization over TCP shall conform to 12.2.4.

3245 **12.3.5 OCF-Content-Format-Version information**

3246 The OCF Content Format Version information used for CoAP serialization over TCP shall conform
3247 to 12.2.5.

3248 **12.3.6 Content-Format policy**

3249 The Content Format policy used for CoAP serialization over TCP shall conform to 12.2.6.

3250 **12.3.7 CRUDN to CoAP response codes**

3251 The CRUDN to CoAP response codes for CoAP serialization over TCP shall conform to 12.2.7.

3252 **12.3.8 CoAP block transfer**

3253 The CoAP block transfer for CoAP serialization over TCP shall conform to clause 6 of
3254 IETF RFC 8323.

3255 **12.3.9 Keep alive (connection health)**

3256 The Device that initiated the CoAP over TCP connection shall send a Ping message as described
3257 in clause 5.4 in IETF RFC 8323. The Device to which the connection was made may send a Ping
3258 message. The recipient of any Ping message shall send a Pong message as described in clause
3259 5.4 in IETF RFC 8323.

3260 Both sides of an established CoAP over TCP connection may send subsequent Ping (and
3261 corresponding Pong) messages.

3262 **12.4 Payload Encoding in CBOR**

3263 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to
3264 JSON from CBOR in accordance with IETF RFC 7049 clause 4 unless otherwise specified in this
3265 clause.

3266 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types
3267 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-
3268 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation
3269 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer
3270 numbers shall be within the closed interval $[-2^{53}, 2^{53}]$. Properties defined as a JSON number
3271 should be encoded as integers whenever possible; if this is not possible Properties defined as a
3272 JSON number should use single-precision if the loss of precision does not affect the quality of
3273 service, otherwise the Property shall use double-precision.

3274 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values
3275 in any position. If a Property defined as a JSON integer is received encoded other than as an
3276 integer, the implementation may reject this encoding using a final response as appropriate for the
3277 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a Property is
3278 defined as a JSON number an implementation shall accept integers, single- and double-precision
3279 floating point.

3280 **13 Security**

3281 The details for handling security and privacy are specified in ISO/IEC 30118-2:2018.

3282
3283
3284
3285

Annex A (normative)

Resource Type definitions

3286 A.1 List of Resource Type definitions

3287 All the clauses in Annex A describe the Resource Types with a RESTful API definition language.
3288 The Resource Type definitions presented in Annex A are formatted for readability, and so may
3289 appear to have extra line breaks. Table A.1 contains the list of defined Core Common Resources
3290 in this document.

3291 **Table A.1 – Alphabetized list of Core Resources**

Friendly Name (informative)	Resource Type (rt)	Clause
Atomic Measurement	"oic.wk.atomicmeasurement"	A.2
Collections	"oic.wk.col"	A.3
Device	"oic.wk.d"	A.4
Discoverable Resource	"oic.wk.res"	A.7
Introspection	"oic.wk.introspection"	A.5
Platform	"oic.wk.p"	A.6

3292 A.2 Atomic Measurement links list representation

3293 A.2.1 Introduction

3294 The oic.if.baseline OCF Interface exposes a representation of the links and
3295 the Common Properties of the Atomic Measurement Resource.
3296

3297 A.2.2 Example URI

3298 /AtomicMeasurementResURI

3299 A.2.3 Resource type

3300 The Resource Type is defined as: "oic.wk.atomicmeasurement".

3301 A.2.4 OpenAPI 2.0 definition

```
3302 {  
3303     "swagger": "2.0",  
3304     "info": {  
3305         "title": "Atomic Measurement links list representation",  
3306         "version": "2019-03-04",  
3307         "license": {  
3308             "name": "OCF Data Model License",  
3309             "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",  
3310             "x-copyright": "Copyright 2018-2019 Open Connectivity Foundation, Inc. All rights reserved."  
3311         },  
3312         "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
3313     },  
3314     "schemes": ["http"],  
3315     "consumes": ["application/json"],  
3316     "produces": ["application/json"],  
3317     "paths": {  
3318         "/AtomicMeasurementResURI?if=oic.if.ll": {  
3319             "get": {  
3320                 "description": "The oic.if.ll OCF Interface exposes a representation  
3321 of the Links",
```

```

3322         "parameters": [
3323     {
3324         "$ref": "#/parameters/interface-all"
3325     }
3326 ],
3327     "responses": {
3328         "200": {
3329             "description": "",
3330             "x-example": [{
3331                 "href": "/temperature",
3332                 "rt": ["oic.r.temperature"],
3333                 "if": ["oic.if.s", "oic.if.baseline"]
3334             },
3335             {
3336                 "href": "/bodylocation",
3337                 "rt": ["oic.r.body.location.temperature"],
3338                 "if": ["oic.if.s", "oic.if.baseline"]
3339             },
3340             {
3341                 "href": "/timestamp",
3342                 "rt": ["oic.r.time.stamp"],
3343                 "if": ["oic.if.s", "oic.if.baseline"]
3344             }
3345         ],
3346         "schema": {
3347             "$ref": "#/definitions/links"
3348         }
3349     }
3350 },
3351     "/AtomicMeasurementResURI?if=oic.if.b": {
3352         "get": {
3353             "description": "The oic.if.b OCF Interface returns data items
3354 retrieved from Resources pointed to by the Links.\n",
3355             "parameters": [
3356     {
3357         "$ref": "#/parameters/interface-all"
3358     }
3359 ],
3360     "responses": {
3361         "200": {
3362             "description": "Normal response, no errors, all
3363 Properties are returned correctly\n",
3364             "x-example": [{
3365                 "href": "/temperature",
3366                 "rep": {
3367                     "temperature": 38,
3368                     "units": "C",
3369                     "range": [25, 45]
3370                 }
3371             },
3372             {
3373                 "href": "/bodylocation",
3374                 "rep": {
3375                     "bloc": "ear"
3376                 }
3377             },
3378             {
3379                 "href": "/timestamp",
3380                 "rep": {
3381                     "timestamp": "2007-04-05T14:30+09:00"
3382                 }
3383             }
3384         ],
3385         "schema": {
3386             "$ref": "#/definitions/batch-retrieve"
3387         }
3388     }
3389 },
3390     "/AtomicMeasurementResURI?if=oic.if.baseline": {

```

```

3393         "get": {
3394             "description": "The oic.if.baseline OCF Interface exposes a
3395 representation of the links and\nthe Common Properties of the Atomic Measurement Resource.\n",
3396             "parameters": [
3397                 {
3398                     "$ref": "#/parameters/interface-all"
3399                 }
3400             ],
3401             "responses": {
3402                 "200": {
3403                     "description": "",
3404                     "x-example": {
3405                         "rt": ["oic.wk.atomicmeasurement"],
3406                         "if": ["oic.if.b", "oic.if.ll",
3407 "oic.if.baseline"],
3408                         "rts": ["oic.r.temperature",
3409 "oic.r.body.location.temperature", "oic.r.time.stamp"],
3410                         "rts-m": ["oic.r.temperature",
3411 "oic.r.body.location.temperature", "oic.r.time.stamp"],
3412                         "links": [{
3413                             "href": "/temperature",
3414                             "rt": ["oic.r.temperature"],
3415                             "if": ["oic.if.s", "oic.if.baseline"]
3416                         },
3417                         {
3418                             "href": "/bodylocation",
3419                             "rt":
3420 ["oic.r.body.location.temperature"],
3421                             "if": ["oic.if.s", "oic.if.baseline"]
3422                         },
3423                         {
3424                             "href": "/timestamp",
3425                             "rt": ["oic.r.time.stamp"],
3426                             "if": ["oic.if.s", "oic.if.baseline"]
3427                         }
3428                     ],
3429                     "schema": {
3430                         "$ref": "#/definitions/baseline"
3431                     }
3432                 }
3433             }
3434         }
3435     },
3436     "parameters": {
3437         "interface-all": {
3438             "in": "query",
3439             "name": "if",
3440             "type": "string",
3441             "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
3442         }
3443     },
3444     "definitions": {
3445         "links": {
3446             "type": "array",
3447             "items": {
3448                 "$ref": "#/definitions/oic.oic-link"
3449             }
3450         },
3451         "batch-retrieve": {
3452             "title": "Collection Batch Retrieve Format (auto merged)",
3453             "minItems": 1,
3454             "items": {
3455                 "additionalProperties": true,
3456                 "properties": {
3457                     "href": {
3458                         "$ref":
3459 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3460 schema.json#/definitions/href"
3461                     },
3462                     "rep": {
3463

```



```

3464         "oneOf": [{
3465             "description": "The response payload from a
3466 single Resource",
3467             "type": "object"
3468         },
3469         {
3470             "description": " The response payload from a
3471 Collection (batch) Resource",
3472             "items": {
3473                 "properties": {
3474                     "anchor": {
3475                         "$ref":
3476 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3477 schema.json#/definitions/anchor"
3478                     },
3479                     "di": {
3480                         "$ref":
3481 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3482 schema.json#/definitions/di"
3483                     },
3484                     "eps": {
3485                         "$ref":
3486 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3487 schema.json#/definitions/eps"
3488                     },
3489                     "href": {
3490                         "$ref":
3491 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3492 schema.json#/definitions/href"
3493                     },
3494                     "if": {
3495                         "description": "The OCF
3496 Interface set supported by this Resource",
3497                         "items": {
3498                             "enum": [
3499                                 "oic.if.baseline",
3500                                 "oic.if.ll",
3501                                 "oic.if.b",
3502                                 "oic.if.rw",
3503                                 "oic.if.r",
3504                                 "oic.if.a",
3505                                 "oic.if.s"],
3506                                 "type": "string"
3507                             }
3508                         },
3509                         "minItems": 1,
3510                         "uniqueItems": true,
3511                         "type": "array"
3512                     },
3513                     "ins": {
3514                         "$ref":
3515 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3516 schema.json#/definitions/ins"
3517                     },
3518                     "p": {
3519                         "$ref":
3520 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3521 schema.json#/definitions/p"
3522                     },
3523                     "rel": {
3524                         "description": "The relation of the target URI
3525 referenced by the Link to the context URI",
3526                         "oneOf": [
3527                             {
3528                                 "$ref":
3529 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3530 schema.json#/definitions/rel_array"
3531                             },
3532                             {
3533                                 "$ref":

```

```

3535 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3536 schema.json#/definitions/rel_string"
3537         }
3538     ],
3539 },
3540         "rt": {
3541             "description":
3542 "Resource Type of the Resource",
3543             "items": {
3544                 "maxLength":
3545 64,
3546                 "type":
3547 "string"
3548             },
3549             "minItems": 1,
3550             "uniqueItems": true,
3551             "type": "array"
3552         },
3553         "title": {
3554             "$ref":
3555 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3556 schema.json#/definitions/title"
3557         },
3558         "type": {
3559             "$ref":
3560 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3561 schema.json#/definitions/type"
3562         }
3563     },
3564     "required": [
3565         "href",
3566         "rt",
3567         "if"
3568     ],
3569     "type": "object"
3570 },
3571 "type": "array"
3572     ]}
3573     },
3574     "required": [
3575         "href",
3576         "rep"
3577     ],
3578     "type": "object"
3579 },
3580 "type": "array"
3581 },
3582 "baseline": {
3583     "properties": {
3584         "links": {
3585             "description": "A set of simple or individual Links.",
3586             "items": {
3587                 "$ref": "#/definitions/oic.oic-link"
3588             },
3589             "type": "array"
3590         },
3591         "n": { "$ref": :
3592 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3593 schema.json#/definitions/n"},
3594         "id": { "$ref": :
3595 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3596 schema.json#/definitions/id"},
3597         "rt": {
3598             "description": "Resource Type of this Resource",
3599             "items": {
3600                 "enum": ["oic.wk.atomicmeasurement"],
3601                 "type": "string",
3602                 "maxLength": 64
3603             },
3604             "minItems": 1,

```

```

3606         "readOnly": true,
3607         "uniqueItems": true,
3608         "type": "array"
3609     },
3610         "rts": {
3611             "description": "An array of Resource Types that are supported
3612 within an array of Links exposed by the Resource",
3613             "items": {
3614                 "maxLength": 64,
3615                 "type": "string"
3616             },
3617             "minItems": 1,
3618             "readOnly": true,
3619             "uniqueItems": true,
3620             "type": "array"
3621         },
3622         "rts-m": {
3623             "description": "An array of Resource Types that are mandatory
3624 to be exposed within an array of Links exposed by the Resource",
3625             "items": {
3626                 "maxLength": 64,
3627                 "type": "string"
3628             },
3629             "minItems": 1,
3630             "readOnly": true,
3631             "uniqueItems": true,
3632             "type": "array"
3633         },
3634         "if": {
3635             "description": "The OCF Interface set supported by this
3636 Resource",
3637             "items": {
3638                 "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"],
3639                 "type": "string"
3640             },
3641             "minItems": 3,
3642             "readOnly": true,
3643             "uniqueItems": true,
3644             "type": "array"
3645         }
3646     },
3647     "type": "object",
3648     "required": [
3649         "rt",
3650         "if",
3651         "links"
3652     ]
3653 },
3654 "oic.oic-link": {
3655     "properties": {
3656         "anchor": {
3657             "$ref":
3658 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3659 schema.json#/definitions/anchor"
3660         },
3661         "di": {
3662             "$ref":
3663 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3664 schema.json#/definitions/di"
3665         },
3666         "eps": {
3667             "$ref":
3668 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3669 schema.json#/definitions/eps"
3670         },
3671         "href": {
3672             "$ref":
3673 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3674 schema.json#/definitions/href"
3675         },
3676         "if": {

```

```

3677         "description": "The OCF Interface set supported by this
3678 Resource",
3679         "items": {
3680             "enum": [
3681                 "oic.if.baseline",
3682                 "oic.if.ll",
3683                 "oic.if.b",
3684                 "oic.if.rw",
3685                 "oic.if.r",
3686                 "oic.if.a",
3687                 "oic.if.s"],
3688             "type": "string"
3689         },
3690         "minItems": 1,
3691         "uniqueItems": true,
3692         "type": "array"
3693     },
3694     "ins": {
3695         "$ref":
3696         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3697         schema.json#/definitions/ins"
3698     },
3699     "p": {
3700         "$ref":
3701         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3702         schema.json#/definitions/p"
3703     },
3704     "rel": {
3705         "description": "The relation of the target URI referenced by the Link to the context URI",
3706         "oneOf": [
3707             {
3708                 "$ref":
3709                 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3710                 schema.json#/definitions/rel_array"
3711             },
3712             {
3713                 "$ref":
3714                 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3715                 schema.json#/definitions/rel_string"
3716             }
3717         ]
3718     },
3719     "rt": {
3720         "description": "Resource Type of the Resource",
3721         "items": {
3722             "maxLength": 64,
3723             "type": "string"
3724         },
3725         "minItems": 1,
3726         "uniqueItems": true,
3727         "type": "array"
3728     },
3729     "title": {
3730         "$ref":
3731         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3732         schema.json#/definitions/title"
3733     },
3734     "type": {
3735         "$ref":
3736         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3737         schema.json#/definitions/type"
3738     }
3739 },
3740 "required": [
3741     "href",
3742     "rt",
3743     "if"
3744 ],
3745 "type": "object"
3746 }
3747 }

```

3748 }
3749

3750 A.2.5 Property definition

3751 Table A.2 defines the Properties that are part of the "oic.wk.atomicmeasurement" Resource Type.

3752
3753

Table A.2 – The Property definitions of the Resource with type "rt" = "oic.wk.atomicmeasurement".

Property name	Value type	Mandatory	Access mode	Description
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
links	array: see schema	Yes	Read Write	A set of simple or individual Links.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource
rts	array: see schema	No	Read Only	An array of Resource Types that are supported within an array of Links exposed by the Resource
rts-m	array: see schema	No	Read Only	An array of Resource Types that are mandatory to be exposed within an array of Links exposed by the Resource
if	array: see schema	Yes	Read Only	The OCF Interface set supported by this Resource
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interface set supported by this Resource
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the

				Link to the context URI
rt	array: see schema	Yes	Read Write	Resource Type of the Resource
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	

3754 **A.2.6 CRUDN behaviour**

3755 Table A.3 defines the CRUDN operations that are supported on the "oic.wk.atomicmeasurement"
3756 Resource Type.

3757 **Table A.3 – The CRUDN operations of the Resource with type "rt" =**
3758 **"oic.wk.atomicmeasurement".**

Create	Read	Update	Delete	Notify
	get			observe

3759 **A.3 Collection**

3760 **A.3.1 Introduction**

3761 Collection Resource Type contains Properties and Links.
3762 The oic.if.baseline OCF Interface exposes a representation of
3763 the Links and the Properties of the Collection Resource itself
3764

3765 **A.3.2 Example URI**

3766 /CollectionResURI

3767 **A.3.3 Resource type**

3768 The Resource Type is defined as: "oic.wk.col".

3769 **A.3.4 OpenAPI 2.0 definition**

```

3770 {
3771   "swagger": "2.0",
3772   "info": {
3773     "title": "Collection",
3774     "version": "2019-03-04",
3775     "license": {
3776       "name": "OCF Data Model License",
3777       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
3778       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
3779     },
3780     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
3781   },
3782   "schemes": [
3783     "http"
3784   ],
3785   "consumes": [
3786     "application/json"
3787   ],
3788   "produces": [
3789     "application/json"
3790   ],
3791   "paths": {
3792     "/CollectionResURI?if=oic.if.ll" : {
3793       "get": {
3794         "description": "Collection Resource Type contains Properties and Links.\n\nThe oic.if.ll OCF

```

```

3795 Interface exposes a representation of the Links\n",
3796     "parameters": [
3797     {
3798         "$ref": "#/parameters/interface-all"
3799     }
3800 ],
3801     "responses": {
3802         "200": {
3803             "description" : "",
3804             "x-example": [
3805                 {
3806                     "href": "/switch",
3807                     "rt": ["oic.r.switch.binary"],
3808                     "if": ["oic.if.a", "oic.if.baseline"],
3809                     "eps": [
3810                         {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3811                         {"ep": "coaps://[fe80::b1d6]:1122"},
3812                         {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
3813                     ]
3814                 },
3815                 {
3816                     "href": "/airFlow",
3817                     "rt": ["oic.r.airflow"],
3818                     "if": ["oic.if.a", "oic.if.baseline"],
3819                     "eps": [
3820                         {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3821                         {"ep": "coaps://[fe80::b1d6]:1122"},
3822                         {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
3823                     ]
3824                 }
3825             ],
3826             "schema": {
3827                 "$ref": "#/definitions/slinks"
3828             }
3829         }
3830     }
3831 },
3832 ],
3833 "/CollectionResURI?if=oic.if.baseline" : {
3834     "get": {
3835         "description": "Collection Resource Type contains Properties and Links.\n\nThe oic.if.baseline
3836 OCF Interface exposes a representation of\n\nthe Links and the Properties of the Collection Resource
3837 itself\n",
3838         "parameters": [
3839         {
3840             "$ref": "#/parameters/interface-all"
3841         }
3842     ],
3843     "responses": {
3844         "200": {
3845             "description" : "",
3846             "x-example": {
3847                 "rt": ["oic.wk.col"],
3848                 "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
3849                 "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
3850                 "rts-m": [ "oic.r.switch.binary" ],
3851                 "links": [
3852                     {
3853                         "href": "/switch",
3854                         "rt": ["oic.r.switch.binary"],
3855                         "if": ["oic.if.a", "oic.if.baseline"],
3856                         "eps": [
3857                             {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3858                             {"ep": "coaps://[fe80::b1d6]:1122"},
3859                             {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
3860                         ]
3861                     },
3862                     {
3863                         "href": "/airFlow",
3864                         "rt": ["oic.r.airflow"],
3865                         "if": ["oic.if.a", "oic.if.baseline"],

```

```

3866         "eps": [
3867             { "ep": "coap://[fe80::b1d6]:1111", "pri": 2 },
3868             { "ep": "coaps://[fe80::b1d6]:1122" },
3869             { "ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3 }
3870         ]
3871     }
3872 ]
3873 },
3874 "schema": {
3875     "$ref": "#/definitions/sbaseline"
3876 }
3877 }
3878 },
3879 ],
3880 "post": {
3881     "description": "Update on Baseline OCF Interface\n",
3882     "parameters": [
3883         {
3884             "$ref": "#/parameters/interface-update"
3885         },
3886         {
3887             "name": "body",
3888             "in": "body",
3889             "required": true,
3890             "schema": {
3891                 "$ref": "#/definitions/sbaseline-update"
3892             }
3893         }
3894     ],
3895     "responses": {
3896         "200": {
3897             "description": "",
3898             "schema": {
3899                 "$ref": "#/definitions/sbaseline"
3900             }
3901         }
3902     }
3903 },
3904 ],
3905 "/CollectionResURI?if=oic.if.b" : {
3906     "get": {
3907         "description": "Collection Resource Type contains Properties and Links.\n\nThe oic.if.b OCF
3908 Interface exposes a composite representation of the\n\nResources pointed to by the Links\n",
3909         "parameters": [
3910             {
3911                 "$ref": "#/parameters/interface-all"
3912             }
3913         ],
3914         "responses": {
3915             "200": {
3916                 "description": "All targets returned OK status",
3917                 "x-example": [
3918                     {
3919                         "href": "/switch",
3920                         "rep": {
3921                             "value": true
3922                         }
3923                     },
3924                     {
3925                         "href": "/airFlow",
3926                         "rep": {
3927                             "direction": "floor",
3928                             "speed": 3
3929                         }
3930                     }
3931                 ],
3932                 "schema": {
3933                     "$ref": "#/definitions/sbatch-retrieve"
3934                 }
3935             },
3936             "404": {

```



```

3937         "description" : "One or more targets did not return an OK status, return a
3938 representation containing returned Properties from the targets that returned OK",
3939         "x-example": [
3940             {
3941                 "href": "/switch",
3942                 "rep": {
3943                     "value": true
3944                 }
3945             },
3946             {
3947                 "schema": {
3948                     "$ref": "#/definitions/sbatch-retrieve"
3949                 }
3950             }
3951         ],
3952     },
3953     "post": {
3954         "description": "Update on Batch OCF Interface\n",
3955         "parameters": [
3956             {
3957                 "$ref": "#/parameters/interface-update"
3958             },
3959             {
3960                 "name": "body",
3961                 "in": "body",
3962                 "required": true,
3963                 "schema": {
3964                     "$ref": "#/definitions/sbatch-update"
3965                 },
3966                 "x-example": [
3967                     {
3968                         "href": "/switch",
3969                         "rep": {
3970                             "value": true
3971                         }
3972                     },
3973                     {
3974                         "href": "/airFlow",
3975                         "rep": {
3976                             "direction": "floor",
3977                             "speed": 3
3978                         }
3979                     }
3980                 ]
3981             }
3982         ],
3983         "responses": {
3984             "200": {
3985                 "description" : "All targets returned OK status, return a representation of the current
3986 state of all targets",
3987                 "x-example": [
3988                     {
3989                         "href": "/switch",
3990                         "rep": {
3991                             "value": true
3992                         }
3993                     },
3994                     {
3995                         "href": "/airFlow",
3996                         "rep": {
3997                             "direction": "demist",
3998                             "speed": 5
3999                         }
4000                     }
4001                 ],
4002                 "schema": {
4003                     "$ref": "#/definitions/sbatch-retrieve"
4004                 }
4005             },
4006             "403": {
4007                 "description" : "One or more targets did not return OK status; return a retrieve

```

```

4008 representation of the current state of all targets in the batch",
4009     "x-example": [
4010         {
4011             "href": "/switch",
4012             "rep": {
4013                 "value": true
4014             }
4015         },
4016         {
4017             "href": "/airFlow",
4018             "rep": {
4019                 "direction": "floor",
4020                 "speed": 3
4021             }
4022         }
4023     ],
4024     "schema": {
4025         "$ref": "#/definitions/sbatch-retrieve"
4026     }
4027 }
4028 }
4029 }
4030 },
4031 },
4032 "parameters": {
4033     "interface-all" : {
4034         "in" : "query",
4035         "name" : "if",
4036         "type" : "string",
4037         "enum" : ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
4038     },
4039     "interface-update" : {
4040         "in" : "query",
4041         "name" : "if",
4042         "type" : "string",
4043         "enum" : ["oic.if.b", "oic.if.baseline"]
4044     }
4045 },
4046 "definitions": {
4047     "sbaseline" : {
4048         "properties": {
4049             "links" : {
4050                 "description": "A set of simple or individual Links.",
4051                 "items": {
4052                     "$ref": "#/definitions/oic.oic-link"
4053                 },
4054                 "type": "array"
4055             },
4056             "n": {
4057                 "$ref" :
4058 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4059 schema.json#/definitions/n"
4060             },
4061             "id": {
4062                 "$ref" :
4063 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4064 schema.json#/definitions/id"
4065             },
4066             "rt": {
4067                 "$ref": "#/definitions/oic.core.rt-col"
4068             },
4069             "rts": {
4070                 "$ref": "#/definitions/oic.core.rt"
4071             },
4072             "rts-m": {
4073                 "$ref": "#/definitions/oic.core.rt"
4074             },
4075             "if": {
4076                 "description": "The OCF Interfaces supported by this Resource",
4077                 "items": {
4078                     "enum": [

```

```

4079         "oic.if.ll",
4080         "oic.if.baseline",
4081         "oic.if.b"
4082     ],
4083         "type": "string",
4084         "maxLength": 64
4085     },
4086     "minItems": 2,
4087     "uniqueItems": true,
4088     "readOnly": true,
4089     "type": "array"
4090 }
4091 },
4092 "additionalProperties": true,
4093 "type": "object",
4094 "required": [
4095     "rt",
4096     "if",
4097     "links"
4098 ]
4099 },
4100 "sbaseline-update": {
4101     "additionalProperties": true
4102 },
4103     "oic.core.rt-col": {
4104         "description": "Resource Type of the Resource",
4105         "items": {
4106             "enum": ["oic.wk.col"],
4107             "type": "string",
4108         },
4109         "maxLength": 64
4110     },
4111     "minItems": 1,
4112     "uniqueItems": true,
4113     "readOnly": true,
4114     "type": "array"
4115 },
4116 "oic.core.rt": {
4117     "description": "Resource Type or set of Resource Types",
4118     "items": {
4119         "type": "string",
4120     },
4121     "minItems": 1,
4122     "uniqueItems": true,
4123     "readOnly": true,
4124     "type": "array"
4125 },
4126 "sbatch-retrieve" : {
4127     "minItems" : 1,
4128     "items" : {
4129         "additionalProperties": true,
4130         "properties": {
4131             "href": {
4132                 "$ref":
4133                 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4134                 schema.json#/definitions/href"
4135             },
4136             "rep": {
4137                 "oneOf": [
4138                     {
4139                         "description": "The response payload from a single Resource",
4140                         "type": "object"
4141                     },
4142                     {
4143                         "description": " The response payload from a Collection (batch) Resource",
4144                         "items": {
4145                             "$ref": "#/definitions/oic.oic-link"
4146                         },
4147                         "type": "array"
4148                     }
4149                 ]

```

```

4150     }
4151   },
4152   "required": [
4153     "href",
4154     "rep"
4155   ],
4156   "type": "object"
4157 },
4158 "type" : "array"
4159 },
4160 "sbatch-update" : {
4161   "title" : "Collection Batch Update Format",
4162   "minItems" : 1,
4163   "items" : {
4164     "$ref": "#/definitions/sbatch-update.item"
4165   },
4166   "type" : "array"
4167 },
4168 "sbatch-update.item" : {
4169   "additionalProperties": true,
4170   "description": "Array of Resource representations to apply to the batch Collection, using href
4171 to indicate which Resource(s) in the batch to update. If the href Property is empty, effectively
4172 making the URI reference to the Collection itself, the representation is to be applied to all
4173 Resources in the batch",
4174   "properties": {
4175     "href": {
4176       "$ref":
4177 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4178 schema.json#/definitions/href"
4179     },
4180     "rep": {
4181       "oneOf": [
4182         {
4183           "description": "The payload for a single Resource",
4184           "type": "object"
4185         },
4186         {
4187           "description": " The payload for a Collection (batch) Resource",
4188           "items": {
4189             "$ref": "#/definitions/oic.oic-link"
4190           },
4191           "type": "array"
4192         }
4193       ]
4194     }
4195   },
4196   "required": [
4197     "href",
4198     "rep"
4199   ],
4200   "type": "object"
4201 },
4202 "slinks" : {
4203   "type" : "array",
4204   "items" : {
4205     "$ref": "#/definitions/oic.oic-link"
4206   }
4207 },
4208 "oic.oic-link": {
4209   "properties": {
4210     "if": {
4211       "description": "The OCF Interfaces supported by the Linked target",
4212       "items": {
4213         "enum": [
4214           "oic.if.baseline",
4215           "oic.if.ll",
4216           "oic.if.b",
4217           "oic.if.rw",
4218           "oic.if.r",
4219           "oic.if.a",
4220           "oic.if.s"

```

```

4221         ],
4222         "type": "string",
4223         "maxLength": 64
4224     },
4225     "minItems": 1,
4226     "uniqueItems": true,
4227     "readOnly": true,
4228     "type": "array"
4229 },
4230 "rt": {
4231     "$ref": "#/definitions/oic.core.rt"
4232 },
4233 "anchor": {
4234     "$ref":
4235 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4236 schema.json#/definitions/anchor"
4237 },
4238 "di": {
4239     "$ref":
4240 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4241 schema.json#/definitions/di"
4242 },
4243 "eps": {
4244     "$ref":
4245 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4246 schema.json#/definitions/eps"
4247 },
4248 "href": {
4249     "$ref":
4250 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4251 schema.json#/definitions/href"
4252 },
4253 "ins": {
4254     "$ref":
4255 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4256 schema.json#/definitions/ins"
4257 },
4258 "p": {
4259     "$ref":
4260 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4261 schema.json#/definitions/p"
4262 },
4263 "rel": {
4264     "$ref":
4265 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4266 schema.json#/definitions/rel_array"
4267 },
4268 "title": {
4269     "$ref":
4270 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4271 schema.json#/definitions/title"
4272 },
4273 "type": {
4274     "$ref":
4275 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4276 schema.json#/definitions/type"
4277 },
4278 "tag-pos-desc": {
4279     "$ref":
4280 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4281 schema.json#/definitions/tag-pos-desc"
4282 },
4283 "tag-pos-rel": {
4284     "$ref":
4285 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4286 schema.json#/definitions/tag-pos-rel"
4287 },
4288 "tag-func-desc": {
4289     "$ref":
4290 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4291 schema.json#/definitions/tag-func-desc"

```

```

4292     }
4293   },
4294   "required": [
4295     "href",
4296     "rt",
4297     "if"
4298   ],
4299   "type": "object"
4300 }
4301 }
4302 }
4303

```

4304 A.3.5 Property definition

4305 Table A.4 defines the Properties that are part of the "oic.wk.col" Resource Type.

4306 **Table A.4 – The Property definitions of the Resource with type "rt" = "oic.wk.col".**

Property name	Value type	Mandatory	Access mode	Description
links	array: see schema	Yes	Read Write	A set of simple or individual Links.
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	multiple types: see schema	Yes	Read Write	
rts	multiple types: see schema	No	Read Write	
rts-m	multiple types: see schema	No	Read Write	
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by the Linked target
rt	multiple types: see schema	Yes	Read Write	
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	

ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
tag-pos-desc	multiple types: see schema	No	Read Write	
tag-pos-rel	multiple types: see schema	No	Read Write	
tag-func-desc	multiple types: see schema	No	Read Write	

4307 **A.3.6 CRUDN behaviour**

4308 Table A.5 defines the CRUDN operations that are supported on the "oic.wk.col" Resource Type.

4309 **Table A.5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col".**

Create	Read	Update	Delete	Notify
	get	post		observe

4310 **A.4 Device**

4311 **A.4.1 Introduction**

4312 Known Resource that is hosted by every Server.

4313 Allows for logical Device specific information to be discovered.

4314

4315 **A.4.2 Well-known URI**

4316 /oic/d

4317 **A.4.3 Resource type**

4318 The Resource Type is defined as: "oic.wk.d".

4319 **A.4.4 OpenAPI 2.0 definition**

```

4320 {
4321   "swagger": "2.0",
4322   "info": {
4323     "title": "Device",
4324     "version": "2019-03-13",
4325     "license": {
4326       "name": "OCF Data Model License",
4327       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
4328       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4329     },
4330     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4331   },
4332   "schemes": [
4333     "http"
4334   ],
4335   "consumes": [
4336     "application/json"
4337   ],

```

```

4338     "produces": [
4339         "application/json"
4340     ],
4341     "paths": {
4342         "/oic/d" : {
4343             "get": {
4344                 "description": "Known Resource that is hosted by every Server.\nAllows for logical Device
specific information to be discovered.\n",
4345                 "parameters": [
4346                     {
4347                         "$ref": "#/parameters/interface"
4348                     }
4349                 ],
4350             },
4351             "responses": {
4352                 "200": {
4353                     "description": "",
4354                     "x-example":
4355                         {
4356                             "n": "Device 1",
4357                             "rt": ["oic.wk.d"],
4358                             "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
4359                             "icv": "ocf.2.0.2",
4360                             "dmv": "ocf.res.1.0.0, ocf.sh.1.0.0",
4361                             "piid": "6F0AAC04-2BB0-468D-B57C-16570A26AE48"
4362                         },
4363                     "schema": {
4364                         "$ref": "#/definitions/Device"
4365                     }
4366                 }
4367             }
4368         }
4369     },
4370 },
4371 "parameters": {
4372     "interface" : {
4373         "in": "query",
4374         "name": "if",
4375         "type": "string",
4376         "enum": ["oic.if.r", "oic.if.baseline"]
4377     }
4378 },
4379 "definitions": {
4380     "Device": {
4381         "properties": {
4382             "rt": {
4383                 "description": "Resource Type of the Resource",
4384                 "items": {
4385                     "type": "string",
4386                     "maxLength": 64
4387                 },
4388                 "minItems": 1,
4389                 "readOnly": true,
4390                 "uniqueItems": true,
4391                 "type": "array"
4392             },
4393             "ld": {
4394                 "description": "Localized Descriptions.",
4395                 "items": {
4396                     "properties": {
4397                         "language": {
4398                             "allOf": [
4399                                 {
4400                                     "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
schema.json#/definitions/language-tag"
4401                                 },
4402                                 {
4403                                     "description": "An RFC 5646 language tag.",
4404                                     "readOnly": true
4405                                 }
4406                             ]
4407                         }
4408                     }

```



```

4409         "value": {
4410             "description": "Device description in the indicated language.",
4411             "maxLength": 64,
4412             "readOnly": true,
4413             "type": "string"
4414         }
4415     },
4416     "type": "object"
4417 },
4418 "minItems": 1,
4419 "readOnly": true,
4420 "type": "array"
4421 },
4422 "piid": {
4423     "allof": [
4424         {
4425             "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4426 schema.json#/definitions/uuid"
4427         },
4428         {
4429             "description": "Protocol independent unique identifier for the Device that is
4430 immutable.",
4431             "readOnly": true
4432         }
4433     ]
4434 },
4435 "di": {
4436     "allof": [
4437         {
4438             "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4439 schema.json#/definitions/uuid"
4440         },
4441         {
4442             "description": "Unique identifier for the Device",
4443             "readOnly": true
4444         }
4445     ]
4446 },
4447 "dmno": {
4448     "description": "Model number as designated by manufacturer.",
4449     "maxLength": 64,
4450     "readOnly": true,
4451     "type": "string"
4452 },
4453 "sv": {
4454     "description": "Software version.",
4455     "maxLength": 64,
4456     "readOnly": true,
4457     "type": "string"
4458 },
4459 "dmn": {
4460     "description": "Manufacturer Name.",
4461     "items": {
4462         "properties": {
4463             "language": {
4464                 "allof": [
4465                     {
4466                         "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4467 schema.json#/definitions/language-tag"
4468                     },
4469                     {
4470                         "description": "An RFC 5646 language tag.",
4471                         "readOnly": true
4472                     }
4473                 ]
4474             },
4475             "value": {
4476                 "description": "Manufacturer name in the indicated language.",
4477                 "maxLength": 64,
4478                 "readOnly": true,
4479                 "type": "string"

```

```

4480     }
4481     },
4482     "type": "object"
4483   },
4484   "minItems": 1,
4485   "readOnly": true,
4486   "type": "array"
4487 },
4488 "icv": {
4489   "description": "The version of the Device",
4490   "maxLength": 64,
4491   "readOnly": true,
4492   "type": "string"
4493 },
4494 "dmv": {
4495   "description": "Specification versions of the Resource and Device Specifications to which
4496 this device data model is implemented",
4497   "maxLength": 256,
4498   "readOnly": true,
4499   "type": "string"
4500 },
4501 "n": {
4502   "$ref" :
4503 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4504 schema.json#/definitions/n"
4505 },
4506 "id": {
4507   "$ref" :
4508 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4509 schema.json#/definitions/id"
4510 },
4511 "if": {
4512   "description": "The OCF Interfaces supported by this Resource",
4513   "items": {
4514     "enum": [
4515       "oic.if.r",
4516       "oic.if.baseline"
4517     ],
4518     "type": "string",
4519     "maxLength": 64
4520   },
4521   "minItems": 2,
4522   "uniqueItems": true,
4523   "readOnly": true,
4524   "type": "array"
4525 },
4526 "econame" : {
4527   "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD.",
4528   "type": "string",
4529   "enum": ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"],
4530   "readOnly": true
4531 },
4532 "ecoversion" : {
4533   "description": "Version of ecosystem that a Bridged Device belongs to. Typical version
4534 string format is like n.n (e.g. 5.0).",
4535   "type": "string",
4536   "maxLength": 64,
4537   "readOnly": true
4538 }
4539 },
4540 "type": "object",
4541 "required": ["n", "di", "icv", "dmv", "piid"]
4542 }
4543 }
4544 }
4545

```

4546 **A.4.5 Property definition**

4547 Table A.6 defines the Properties that are part of the "oic.wk.d" Resource Type.

Table A.6 – The Property definitions of the Resource with type "rt" = "oic.wk.d".

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
ld	array: see schema	No	Read Only	Localized Descriptions.
piid	multiple types: see schema	Yes	Read Write	
di	multiple types: see schema	Yes	Read Write	
dmno	string	No	Read Only	Model number as designated by manufacturer.
sv	string	No	Read Only	Software version.
dmn	array: see schema	No	Read Only	Manufacturer Name.
icv	string	Yes	Read Only	The version of the Device
dmv	string	Yes	Read Only	Specification versions of the Resource and Device Specifications to which this device data model is implemented
n	multiple types: see schema	Yes	Read Write	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource
econame	string	No	Read Only	Ecosystem Name of the Bridged Device which is exposed by this VOD.
ecoversion	string	No	Read Only	Version of ecosystem that a Bridged Device belongs to. Typical version string format is like n.n (e.g. 5.0).

A.4.6 CRUDN behaviour

4550 Table A.7 defines the CRUDN operations that are supported on the "oic.wk.d" Resource Type.

4551 **Table A.7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d".**

Create	Read	Update	Delete	Notify
	get			observe

4552 **A.5 Introspection Resource**

4553 **A.5.1 Introduction**

4554 This Resource provides the means to get the Introspection Device Data (IDD) specifying all the
4555 OCF Endpoints of the Device.

4556 The url hosted by this Resource is either a local or an external url.

4557

4558 **A.5.2 Well-known URI**

4559 /IntrospectionResURI

4560 **A.5.3 Resource type**

4561 The Resource Type is defined as: "oic.wk.introspection".

4562 **A.5.4 OpenAPI 2.0 definition**

```
4563 {  
4564   "swagger": "2.0",  
4565   "info": {  
4566     "title": "Introspection Resource",  
4567     "version": "2019-03-04",  
4568     "license": {  
4569       "name": "OCF Data Model License",  
4570       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",  
4571       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."  
4572     },  
4573     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
4574   },  
4575   "schemes": [  
4576     "http"  
4577   ],  
4578   "consumes": [  
4579     "application/json"  
4580   ],  
4581   "produces": [  
4582     "application/json"  
4583   ],  
4584   "paths": {  
4585     "/IntrospectionResURI": {  
4586       "get": {  
4587         "description": "This Resource provides the means to get the Introspection Device Data (IDD)  
4588 specifying all the OCF Endpoints of the Device.\n\nThe url hosted by this Resource is either a local  
4589 or an external url.\n\n",  
4590         "parameters": [  
4591           {  
4592             "$ref": "#/parameters/interface"  
4593           }  
4594         ],  
4595         "responses": {  
4596           "200": {  
4597             "description": "",  
4598             "x-example": {  
4599               "rt": ["oic.wk.introspection"],  
4600               "urlInfo": [  
4601                 {  
4602                   "content-type": "application/cbor",  
4603                   "protocol": "coap",  
4604                   "url": "coap://[fe80::1]:1234/IntrospectionExampleURI"  
4605                 }  
4606               ]  
4607             },  
4608             "schema": {  
4609               "$ref": "#/definitions/oic.wk.introspectionInfo"  
4610             }  
4611           }  
4612         }  
4613       }  
4614     }  
4615   }  
4616 }
```

```

4613     }
4614   }
4615 },
4616 "parameters": {
4617   "interface": {
4618     "in": "query",
4619     "name": "if",
4620     "type": "string",
4621     "enum": ["oic.if.r", "oic.if.baseline"]
4622   }
4623 },
4624 "definitions": {
4625   "oic.wk.introspectionInfo": {
4626     "properties": {
4627       "rt": {
4628         "description": "Resource Type of the Resource",
4629         "items": {
4630           "enum": ["oic.wk.introspection"],
4631           "type": "string",
4632           "maxLength": 64
4633         },
4634         "minItems": 1,
4635         "readOnly": true,
4636         "uniqueItems": true,
4637         "type": "array"
4638       },
4639       "n": {
4640         "$ref":
4641 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4642 schema.json#/definitions/n"
4643       },
4644       "urlInfo": {
4645         "description": "Information on the location of the Introspection Device Data (IDD).",
4646         "items": {
4647           "properties": {
4648             "content-type": {
4649               "default": "application/cbor",
4650               "description": "content-type of the Introspection Device Data",
4651               "enum": [
4652                 "application/json",
4653                 "application/cbor"
4654               ],
4655               "type": "string"
4656             },
4657             "protocol": {
4658               "description": "Identifier for the protocol to be used to obtain the Introspection
4659 Device Data",
4660               "enum": [
4661                 "coap",
4662                 "coaps",
4663                 "http",
4664                 "https",
4665                 "coap+tcp",
4666                 "coaps+tcp"
4667               ],
4668               "type": "string"
4669             },
4670             "url": {
4671               "description": "The URL of the Introspection Device Data.",
4672               "format": "uri",
4673               "type": "string"
4674             },
4675             "version": {
4676               "default": 1,
4677               "description": "The version of the Introspection Device Data that can be
4678 downloaded",
4679               "enum": [
4680                 1
4681               ],
4682               "type": "integer"
4683             }

```

```

4684     },
4685     "required": [
4686         "url",
4687         "protocol"
4688     ],
4689     "type": "object"
4690 },
4691 "minItems": 1,
4692 "readOnly": true,
4693 "type": "array"
4694 },
4695 "id": {
4696     "$ref":
4697 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4698 schema.json#/definitions/id"
4699 },
4700 "if": {
4701     "description": "The OCF Interfaces supported by this Resource",
4702     "items": {
4703         "enum": [
4704             "oic.if.r",
4705             "oic.if.baseline"
4706         ],
4707         "type": "string",
4708         "maxLength": 64
4709     },
4710     "minItems": 2,
4711     "readOnly": true,
4712     "uniqueItems": true,
4713     "type": "array"
4714 },
4715 },
4716 "type" : "object",
4717 "required": ["urlInfo"]
4718 }
4719 }
4720 }
4721

```

4722 **A.5.5 Property definition**

4723 Table A.8 defines the Properties that are part of the "oic.wk.introspection" Resource Type.

4724 **Table A.8 – The Property definitions of the Resource with type "rt" =**
4725 **"oic.wk.introspection".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
n	multiple types: see schema	No	Read Write	
urlInfo	array: see schema	Yes	Read Only	Information on the location of the Introspection Device Data (IDD).
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource

4726 **A.5.6 CRUDN behaviour**

4727 Table A.9 defines the CRUDN operations that are supported on the "oic.wk.introspection" Resource
4728 Type.

4729 **Table A.9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".**

Create	Read	Update	Delete	Notify
	get			observe

4730 **A.6 Platform**

4731 **A.6.1 Introduction**

4732 Known Resource that is defines the Platform on which an Server is hosted.
 4733 Allows for Platform specific information to be discovered.
 4734

4735 **A.6.2 Well-known URI**

4736 /oic/p

4737 **A.6.3 Resource type**

4738 The Resource Type is defined as: "oic.wk.p".

4739 **A.6.4 OpenAPI 2.0 definition**

```

4740 {
4741   "swagger": "2.0",
4742   "info": {
4743     "title": "Platform",
4744     "version": "2019-03-04",
4745     "license": {
4746       "name": "OCF Data Model License",
4747       "url":
4748         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
4749         CENSE.md",
4750       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4751     },
4752     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4753   },
4754   "schemes": ["http"],
4755   "consumes": ["application/json"],
4756   "produces": ["application/json"],
4757   "paths": {
4758     "/oic/p" : {
4759       "get": {
4760         "description": "Known Resource that is defines the Platform on which an Server is
4761         hosted.\nAllows for Platform specific information to be discovered.\n",
4762         "parameters": [
4763           { "$ref": "#/parameters/interface" }
4764         ],
4765         "responses": {
4766           "200": {
4767             "description" : "",
4768             "x-example": {
4769               "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
4770               "rt": ["oic.wk.p"],
4771               "mnmn": "Acme, Inc"
4772             },
4773             "schema": { "$ref": "#/definitions/Platform" }
4774           }
4775         }
4776       }
4777     }
4778   },
4779   "parameters": {
4780     "interface" : {
4781       "in" : "query",
4782       "name" : "if",
4783       "type" : "string",
4784       "enum" : ["oic.if.r", "oic.if.baseline"]
    }
  }

```

```

4785     }
4786   },
4787   "definitions": {
4788     "Platform": {
4789       "properties": {
4790         "rt": {
4791           "description": "Resource Type of the Resource",
4792           "items": {
4793             "enum": ["oic.wk.p"],
4794             "type": "string",
4795             "maxLength": 64
4796           },
4797           "minItems": 1,
4798           "uniqueItems": true,
4799           "readOnly": true,
4800           "type": "array"
4801         },
4802         "pi": {
4803           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4804 9]{12}$",
4805           "type": "string",
4806           "description": "Platform Identifier",
4807           "readOnly": true
4808         },
4809         "mnfv": {
4810           "description": "Manufacturer's firmware version",
4811           "maxLength": 64,
4812           "readOnly": true,
4813           "type": "string"
4814         },
4815         "vid": {
4816           "description": "Manufacturer's defined information for the Platform. The content is
4817 freeform, with population rules up to the manufacturer",
4818           "maxLength": 64,
4819           "readOnly": true,
4820           "type": "string"
4821         },
4822         "mnmn": {
4823           "description": "Manufacturer name",
4824           "maxLength": 64,
4825           "readOnly": true,
4826           "type": "string"
4827         },
4828         "mnmno": {
4829           "description": "Model number as designated by the manufacturer",
4830           "maxLength": 64,
4831           "readOnly": true,
4832           "type": "string"
4833         },
4834         "mnhw": {
4835           "description": "Platform Hardware Version",
4836           "maxLength": 64,
4837           "readOnly": true,
4838           "type": "string"
4839         },
4840         "mnos": {
4841           "description": "Platform Resident OS Version",
4842           "maxLength": 64,
4843           "readOnly": true,
4844           "type": "string"
4845         },
4846         "mndt": {
4847           "pattern": "^[0-9]{4}-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$",
4848           "type": "string",
4849           "description": "Manufacturing Date.",
4850           "readOnly": true
4851         },
4852         "id": {
4853           "$ref":
4854 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4855 schema.json#/definitions/id"

```



```

4856     },
4857     "mnsi" : {
4858         "description": "Manufacturer's Support Information URL",
4859         "format": "uri",
4860         "maxLength": 256,
4861         "readOnly": true,
4862         "type": "string"
4863     },
4864     "mpv" : {
4865         "description": "Platform Version",
4866         "maxLength": 64,
4867         "readOnly": true,
4868         "type": "string"
4869     },
4870     "st" : {
4871         "description": "The date-time format pattern according to IETF RFC 3339.",
4872         "format": "date-time",
4873         "readOnly": true,
4874         "type": "string"
4875     },
4876     "n" : {
4877         "$ref":
4878         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4879         schema.json#/definitions/n"
4880     },
4881     "mnm" : {
4882         "description": "Manufacturer's URL",
4883         "format": "uri",
4884         "maxLength": 256,
4885         "readOnly": true,
4886         "type": "string"
4887     },
4888     "mnsel" : {
4889         "description": "Serial number as designated by the manufacturer",
4890         "maxLength": 64,
4891         "readOnly": true,
4892         "type": "string"
4893     },
4894     "if" : {
4895         "description": "The OCF Interfaces supported by this Resource",
4896         "items": {
4897             "enum": [
4898                 "oic.if.r",
4899                 "oic.if.baseline"
4900             ],
4901             "type": "string",
4902             "maxLength": 64
4903         },
4904         "minItems": 2,
4905         "readOnly": true,
4906         "uniqueItems": true,
4907         "type": "array"
4908     },
4909     "mnct" : {
4910         "description": "An array of integers and each integer indicates the network connectivity
4911         type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-
4912         mib/ianaiftype-mib, e.g., [71, 259] which represents Wi-Fi and Zigbee.",
4913         "items": {
4914             "type": "integer",
4915             "minimum": 1,
4916             "description": "The network connectivity type based on IANAIfType value as defined by:
4917         https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib."
4918         },
4919         "minItems": 1,
4920         "readOnly": true,
4921         "type": "array"
4922     }
4923 },
4924 "type" : "object",
4925 "required": ["pi", "mnm"]
4926 }

```

4927 }
 4928 }
 4929 }

4930 **A.6.5 Property definition**

4931 Table A.10 defines the Properties that are part of the "oic.wk.p" Resource Type.

4932 **Table A.10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
pi	string	Yes	Read Only	Platform Identifier
mnfv	string	No	Read Only	Manufacturer's firmware version
vid	string	No	Read Only	Manufacturer's defined information for the Platform. The content is freeform, with population rules up to the manufacturer
mnmn	string	Yes	Read Only	Manufacturer name
mnmo	string	No	Read Only	Model number as designated by the manufacturer
mnhw	string	No	Read Only	Platform Hardware Version
mnos	string	No	Read Only	Platform Resident OS Version
mndt	string	No	Read Only	Manufacturing Date.
id	multiple types: see schema	No	Read Write	
mnsi	string	No	Read Only	Manufacturer's Support Information URL
mnpv	string	No	Read Only	Platform Version
st	string	No	Read Only	The date-time format pattern according to IETF RFC 3339.
n	multiple types: see schema	No	Read Write	
mnml	string	No	Read Only	Manufacturer's URL
mnsel	string	No	Read Only	Serial number as designated by the manufacturer
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource
mnct	array: see schema	No	Read Only	An array of integers and each integer indicates the network connectivity type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib , e.g., [71, 259] which represents Wi-Fi and Zigbee.

4933 **A.6.6 CRUDN behaviour**

4934 Table A.11 defines the CRUDN operations that are supported on the "oic.wk.p" Resource Type.

4935 **Table A.11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p".**

Create	Read	Update	Delete	Notify
	get			observe

4936 **A.7 Discoverable Resources**

4937 **A.7.1 Introduction**

4938 Baseline representation of /oic/res; list of discoverable Resources

4939

4940 **A.7.2 Well-known URI**

4941 /oic/res

4942 **A.7.3 Resource type**

4943 **A.7.4 OpenAPI 2.0 definition**

```
4944 {
4945   "swagger": "2.0",
4946   "info": {
4947     "title": "Discoverable Resources",
4948     "version": "2019-03-13",
4949     "license": {
4950       "name": "OCF Data Model License",
4951       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
4952       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4953     },
4954     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4955   },
4956   "schemes": [
4957     "http"
4958   ],
4959   "consumes": [
4960     "application/json"
4961   ],
4962   "produces": [
4963     "application/json"
4964   ],
4965   "paths": {
4966     "/oic/res?if=oic.if.ll": {
4967       "get": {
4968         "description": "Links list representation of /oic/res; list of discoverable Resources\n",
4969         "parameters": [
4970           {
4971             "$ref": "#/parameters/interface-all"
4972           }
4973         ],
4974         "responses": {
4975           "200": {
4976             "description": "",
4977             "x-example": [
4978               {
4979                 "href": "/humidity",
4980                 "rt": ["oic.r.humidity"],
4981                 "if": ["oic.if.s", "oic.if.baseline"],
4982                 "p": {"bm": 3},
4983                 "eps": [
4984                   {"ep": "coaps://[fe80::bld6]:1111", "pri": 2},
4985                   {"ep": "coaps://[fe80::bld6]:1122"},
4986                   {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
4987                 ]
4988               },
4989               {
4990                 "href": "/temperature",
4991                 "rt": ["oic.r.temperature"],
4992                 "if": ["oic.if.s", "oic.if.baseline"],
4993                 "p": {"bm": 3},
4994                 "eps": [
4995                   {"ep": "coaps://[[2001:db8:a::123]:2222"}
4996                 ]
4997               }
4998             ]
4999           }
5000         }
5001       }
5002     }
5003   }
5004 }
```

```

4999         "schema": {
5000             "$ref": "#/definitions/slinklist"
5001         }
5002     }
5003 }
5004 }
5005 },
5006 "/oic/res?if=oic.if.baseline": {
5007     "get": {
5008         "description": "Baseline representation of /oic/res; list of discoverable Resources\n",
5009         "parameters": [
5010             {
5011                 "$ref": "#/parameters/interface-all"
5012             }
5013         ],
5014         "responses": {
5015             "200": {
5016                 "description": "",
5017                 "x-example": [
5018                     {
5019                         "rt": ["oic.wk.res"],
5020                         "if": ["oic.if.ll", "oic.if.baseline"],
5021                         "links": [
5022                             {
5023                                 "href": "/humidity",
5024                                 "rt": ["oic.r.humidity"],
5025                                 "if": ["oic.if.s", "oic.if.baseline"],
5026                                 "p": {"bm": 3},
5027                                 "eps": [
5028                                     {"ep": "coaps://[fe80:b1d6]:1111", "pri": 2},
5029                                     {"ep": "coaps://[fe80:b1d6]:1122"},
5030                                     {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5031                                 ]
5032                             },
5033                             {
5034                                 "href": "/temperature",
5035                                 "rt": ["oic.r.temperature"],
5036                                 "if": ["oic.if.s", "oic.if.baseline"],
5037                                 "p": {"bm": 3},
5038                                 "eps": [
5039                                     {"ep": "coaps://[[2001:db8:a::123]:2222"}
5040                                 ]
5041                             }
5042                         ]
5043                     }
5044                 ],
5045                 "schema": {
5046                     "$ref": "#/definitions/sbaseline"
5047                 }
5048             }
5049         }
5050     }
5051 },
5052 },
5053 "parameters": {
5054     "interface-all": {
5055         "in": "query",
5056         "name": "if",
5057         "type": "string",
5058         "enum": ["oic.if.ll", "oic.if.baseline"]
5059     }
5060 },
5061 "definitions": {
5062     "oic.oic-link": {
5063         "type": "object",
5064         "properties": {
5065             "anchor": {
5066                 "$ref":
5067 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5068 schema.json#/definitions/anchor"
5069             }

```

```

5070         "di": {
5071             "$ref":
5072 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5073 schema.json#/definitions/di"
5074         },
5075         "eps": {
5076             "$ref":
5077 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5078 schema.json#/definitions/eps"
5079         },
5080         "href": {
5081             "$ref":
5082 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5083 schema.json#/definitions/href"
5084         },
5085         "if": {
5086             "description": "The OCF Interfaces supported by the Linked Resource",
5087             "items": {
5088                 "enum": [
5089                     "oic.if.baseline",
5090                     "oic.if.ll",
5091                     "oic.if.b",
5092                     "oic.if.rw",
5093                     "oic.if.r",
5094                     "oic.if.a",
5095                     "oic.if.s"
5096                 ],
5097                 "type": "string",
5098                 "maxLength": 64
5099             },
5100             "minItems": 1,
5101             "uniqueItems": true,
5102             "type": "array"
5103         },
5104         "ins": {
5105             "$ref":
5106 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5107 schema.json#/definitions/ins"
5108         },
5109         "p": {
5110             "$ref":
5111 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5112 schema.json#/definitions/p"
5113         },
5114         "rel": {
5115             "description": "The relation of the target URI referenced by the Link to the context URI",
5116             "oneOf": [
5117                 {
5118                     "$ref":
5119 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5120 schema.json#/definitions/rel_array"
5121                 },
5122                 {
5123                     "$ref":
5124 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5125 schema.json#/definitions/rel_string"
5126                 }
5127             ]
5128         },
5129         "rt": {
5130             "description": "Resource Type of the Linked Resource",
5131             "items": {
5132                 "maxLength": 64,
5133                 "type": "string"
5134             },
5135             "minItems": 1,
5136             "uniqueItems": true,
5137             "type": "array"
5138         },
5139         "title": {
5140             "$ref":

```

```

5141 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5142 schema.json#/definitions/title"
5143   },
5144   "type": {
5145     "$ref":
5146 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5147 schema.json#/definitions/type"
5148   },
5149   "tag-pos-desc": {
5150     "$ref":
5151 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5152 schema.json#/definitions/tag-pos-desc"
5153   },
5154   "tag-pos-rel": {
5155     "$ref":
5156 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5157 schema.json#/definitions/tag-pos-rel"
5158   },
5159   "tag-func-desc": {
5160     "$ref":
5161 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5162 schema.json#/definitions/tag-func-desc"
5163   }
5164 },
5165 "required": [
5166   "href",
5167   "rt",
5168   "if"
5169 ]
5170 },
5171 "slinklist": {
5172   "type": "array",
5173   "readOnly": true,
5174   "items": {
5175     "$ref": "#/definitions/oic.oic-link"
5176   }
5177 },
5178 "sbaseline": {
5179   "type": "array",
5180   "minItems": 1,
5181   "maxItems": 1,
5182   "items": {
5183     "type": "object",
5184     "properties": {
5185       "n": {
5186         "$ref":
5187 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5188 schema.json#/definitions/n"
5189       },
5190       "id": {
5191         "$ref":
5192 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5193 schema.json#/definitions/id"
5194       },
5195       "rt": {
5196         "description": "Resource Type of this Resource",
5197         "items": {
5198           "enum": ["oic.wk.res"],
5199           "type": "string",
5200           "maxLength": 64
5201         },
5202         "minItems": 1,
5203         "readOnly": true,
5204         "uniqueItems": true,
5205         "type": "array"
5206       },
5207       "if": {
5208         "description": "The OCF Interfaces supported by this Resource",
5209         "items": {
5210           "enum": [
5211             "oic.if.ll",

```

```

5212         "oic.if.baseline"
5213     ],
5214     "type": "string",
5215     "maxLength": 64
5216 },
5217     "minItems": 2,
5218     "readOnly": true,
5219     "uniqueItems": true,
5220     "type": "array"
5221 },
5222     "links": {
5223         "type": "array",
5224         "items": {
5225             "$ref": "#/definitions/oic.oic-link"
5226         }
5227     }
5228 },
5229 ],
5230     "required": [
5231         "rt",
5232         "if",
5233         "links"
5234     ]
5235 }
5236 }
5237 }
5238

```

5239 **A.7.5 Property definition**

5240 Table A.12 defines the Properties that are part of the "None" Resource Type.

5241 **Table A.12 – The Property definitions of the Resource with type "rt" = "None".**

Property name	Value type	Mandatory	Access mode	Description
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the Linked Resource
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI
rt	array: see schema	Yes	Read Write	Resource Type of the Linked Resource
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	

tag-pos-desc	multiple types: see schema	No	Read Write	
tag-pos-rel	multiple types: see schema	No	Read Write	
tag-func-desc	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource
links	array: see schema	Yes	Read Write	

5242 **A.7.6 CRUDN behaviour**

5243 Table A.13 defines the CRUDN operations that are supported on the "None" Resource Type.

5244 **Table A.13 – The CRUDN operations of the Resource with type "rt" = "None".**

Create	Read	Update	Delete	Notify
	get			observe

5245

5246

5247
5248
5249
5250

Annex B (informative)

OpenAPI 2.0 Schema Extension

5251 **B.1 OpenAPI 2.0 Schema Reference**

5252 OpenAPI 2.0 does not support allOf and anyOf JSON schema validation constructs; this document
5253 has extended the underlying OpenAPI 2.0 schema to enable these, all OpenAPI 2.0 files are valid
5254 against the extended schema. Reference the following location for a copy of the extended schema:

5255 – <https://github.com/openconnectivityfoundation/OCFswagger2.0-schema>

5256 **B.2 OpenAPI 2.0 Introspection empty file**

5257 Reference the following location for a copy of an empty OpenAPI 2.0 file:

5258 – [https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-
5259 examples/introspection-empty.txt](https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-examples/introspection-empty.txt)

5260
5261
5262
5263

Annex C (normative)

Semantic Tag enumeration support

5264 C.1 Introduction

5265 This Annex defines the enumerations that are applicable to defined Semantic Tags.

5266 C.2 "tag-pos-desc" supported enumeration

5267 Figure C.1 defines the enumeration from which a value populated within an instance of the "tag-
5268 pos-desc" Semantic Tag is taken.

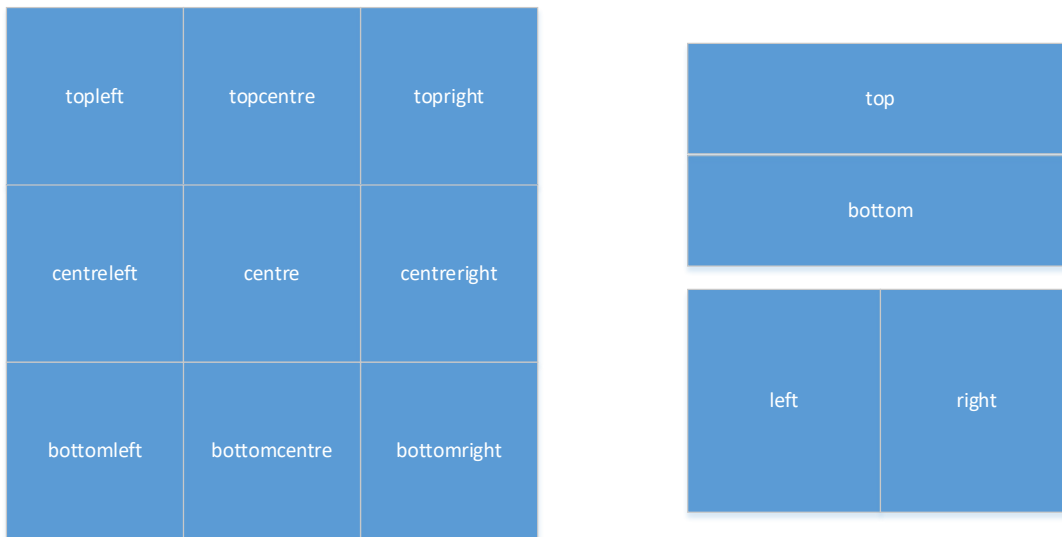
```
"pos-descriptions": {  
  "enum":  
  [ "unknown", "top", "bottom", "left", "right", "centre", "topleft", "bottomleft", "centreleft",  
    "centreright", "bottomright", "topright", "topcentre", "bottomcentre" ]  
}
```

5269

Figure C.1 – Enumeration for "tag-pos-desc" Semantic Tag

5270

5271 Figure C.2 provides an illustrative representation of the definition of the values that can be
5272 represented within an instance of "tag-pos-desc".



5273

5274

Figure C.2 – Definition of "tag-pos-desc" Semantic Tag values

5275

5276

Bibliography

- 5277 [1] OCF Core - Optional, Information technology – Open Connectivity Foundation (OCF)
5278 Specification – Part X: Core - Optional specification
5279 Latest version available at:
5280 https://openconnectivity.org/specs/OCF_Core_Optional_Specification.pdf
- 5281 [2] OCF Easy Wi-Fi Setup, Information technology – Open Connectivity Foundation (OCF)
5282 Specification – Part 7: Wi-Fi Easy Setup specification
5283 Latest version available at: [https://openconnectivity.org/specs/OCF_Wi-](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)
5284 [Fi_Easy_Setup_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)
- 5285