



DRAFT

OCF CORE
SPECIFICATION
V1.0.0
Part 1

Open Connectivity Foundation (OCF)
admin@openconnectivity.org

Legal Disclaimer

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

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

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

Copyright © 2016-2017 Open Connectivity Foundation, Inc. All rights reserved.

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

CONTENTS

26

27

28	1	Scope	13
29	2	Normative references	13
30	3	Terms, definitions, symbols and abbreviations	16
31	3.1	Terms and definitions	16
32	3.2	Symbols and abbreviations	18
33	3.3	Conventions	20
34	3.4	Data types	20
35	4	Document conventions and organization	21
36	5	Architecture.....	22
37	5.1	Overview	22
38	5.2	Principle	22
39	5.3	Functional block diagram.....	24
40	5.4	Framework	25
41	5.5	Example Scenario with roles.....	25
42	5.6	Example Scenario: Bridging to Non- OCF ecosystem.....	26
43	6	Identification and addressing.....	27
44	6.1	Introduction	27
45	6.2	Identification.....	28
46	6.2.1	Resource identification and addressing	28
47	6.3	Namespace:	29
48	6.4	Network addressing	29
49	7	Resource model	29
50	7.1	Introduction	29
51	7.2	Resource.....	30
52	7.3	Property	31
53	7.3.1	Introduction	31
54	7.3.2	Common Properties.....	32
55	7.4	Resource Type	33
56	7.4.1	Introduction	33
57	7.4.2	Resource Type Property.....	34
58	7.4.3	Resource Type definition.....	34
59	7.4.4	Multi-value "rt" Resource.....	35
60	7.5	Device Type	36
61	7.6	Interface	36
62	7.6.1	Introduction	36
63	7.6.2	Interface Property	37
64	7.6.3	Interface methods	37
65	7.7	Resource representation	48
66	7.8	Structure	49
67	7.8.1	Introduction	49
68	7.8.2	Resource Relationships.....	49

69	7.8.3	Collections	54
70	7.9	Third (3 rd) party specified extensions	57
71	8	CRUDN	58
72	8.1	Overview	58
73	8.2	CREATE	59
74	8.2.1	CREATE request	59
75	8.2.2	Processing by the Server	59
76	8.2.3	CREATE response	59
77	8.3	RETRIEVE	60
78	8.3.1	RETRIEVE request.....	60
79	8.3.2	Processing by the Server	60
80	8.3.3	RETRIEVE response	60
81	8.4	UPDATE	61
82	8.4.1	UPDATE request	61
83	8.4.2	Processing by the Server	61
84	8.4.3	UPDATE response	61
85	8.5	DELETE	62
86	8.5.1	DELETE request	62
87	8.5.2	Processing by the Server	62
88	8.5.3	DELETE response.....	62
89	8.6	NOTIFY	63
90	9	Network and connectivity	63
91	9.1	Introduction	63
92	9.2	Architecture	63
93	9.3	• A node may translate and route messaging between IPv6 and non-IPv6 networks.IPv6 network layer requirements.....	65
94			
95	9.3.1	Introduction	65
96	9.3.2	IPv6 node requirements	65
97	10	Endpoint	65
98	10.1	Endpoint definition.....	65
99	10.2	Endpoint information.....	66
100	10.2.1	Introduction	66
101	10.2.2	“ep”	66
102	10.2.3	“pri”	67
103	10.2.4	Endpoint information in "eps" Parameter	67
104	10.3	Endpoint discovery	67
105	10.3.1	Introduction	67
106	10.3.2	Implicit discovery.....	67
107	10.3.3	Explicit discovery with /oic/res response.....	68
108	10.4	CoAP based Endpoint discovery	69
109	11	Functional interactions	70
110	11.1	Introduction	70
111	11.2	Onboarding, Provisioning, and Configuration	70
112	11.3	Resource discovery	72

113	11.3.1	Introduction	72
114	11.3.2	Resource based discovery: mechanisms	72
115	11.3.3	Resource based discovery: Information publication process	74
116	11.3.4	Resource based discovery: Finding information	74
117	11.3.5	Resource discovery using /oic/res	81
118	11.3.6	Resource directory (RD) based discovery.....	83
119	11.4	Notification	92
120	11.4.1	Overview	92
121	11.4.2	Observe	92
122	11.5	Device management	94
123	11.5.1	Overview	94
124	11.5.2	Diagnostics and maintenance	94
125	11.6	Scenes	95
126	11.6.1	Introduction	95
127	11.6.2	Scenes	95
128	11.6.3	Security considerations	99
129	11.7	Icons	99
130	11.7.1	Overview	99
131	11.7.2	Resource.....	100
132	11.8	Introspection.....	100
133	11.8.1	Overview	100
134	11.8.2	Usage of introspection.....	102
135	12	Messaging.....	103
136	12.1	Introduction	103
137	12.2	Mapping of CRUDN to CoAP	104
138	12.2.1	Overview	104
139	12.2.2	URIs.....	104
140	12.2.3	CoAP method with request and response	104
141	12.2.4	Content-Format negotiation	106
142	12.2.5	Content-Format Version information	107
143	12.2.6	Content-Format policy	107
144	12.2.7	CRUDN to CoAP response codes	108
145	12.2.8	CoAP block transfer	108
146	12.3	CoAP serialization over TCP	109
147	12.4	Payload Encoding in CBOR	110
148	13	Security.....	110
149	Annex A (informative)	Operation Examples.....	112
150	A.1	Introduction	112
151	A.2	When at home: From smartphone turn on a single light	112
152	A.3	GroupAction execution	113
153	A.4	When garage door opens, turn on lights in hall; also notify smartphone	113
154	A.5	Device management	113
155	Annex B (informative)	OCF interaction scenarios and deployment models	115
156	B.1	OCF interaction scenarios	115

157	B.2	Deployment model.....	116
158	Annex C (informative)	Other Resource Models and OCF Mapping.....	118
159	C.1	Multiple resource models.....	118
160	C.2	OCF approach for support of multiple resource models.....	118
161	C.3	Resource model indication.....	119
162	C.4	An Example Profile (IPSO profile).....	119
163	C.4.1	Conceptual equivalence.....	119
164	Annex D (normative)	Resource Type definitions.....	122
165	D.1	List of Resource Type definitions.....	122
166	D.2	OCF Collection.....	123
167	D.2.1	Introduction.....	123
168	D.2.2	Example URI.....	123
169	D.2.3	Resource Type.....	123
170	D.2.4	RAML Definition.....	123
171	D.2.5	Property Definition.....	127
172	D.2.6	CRUDN behavior.....	129
173	D.2.7	Referenced JSON schemas.....	129
174	D.2.8	oic.oic-link-schema.json.....	129
175	D.3	OIC Device Configuration.....	131
176	D.3.1	Introduction.....	131
177	D.3.2	Example URI.....	131
178	D.3.3	Resource Type.....	131
179	D.3.4	RAML Definition.....	131
180	D.3.5	Property Definition.....	135
181	D.3.6	CRUDN behavior.....	135
182	D.4	OIC Platform Configuration.....	136
183	D.4.1	Introduction.....	136
184	D.4.2	Example URI.....	136
185	D.4.3	Resource Type.....	136
186	D.4.4	RAML Definition.....	136
187	D.4.5	Property Definition.....	139
188	D.4.6	CRUDN behavior.....	139
189	D.5	Device.....	139
190	D.5.1	Introduction.....	139
191	D.5.2	Wellknown URI.....	139
192	D.5.3	Resource Type.....	139
193	D.5.4	RAML Definition.....	139
194	D.5.5	Property Definition.....	141
195	D.5.6	CRUDN behavior.....	142
196	D.6	Maintenance.....	142
197	D.6.1	Introduction.....	142
198	D.6.2	Wellknown URI.....	142
199	D.6.3	Resource Type.....	142
200	D.6.4	RAML Definition.....	142

201	D.6.5	Property Definition	144
202	D.6.6	CRUDN behavior.....	145
203	D.7	Platform.....	145
204	D.7.1	Introduction	145
205	D.7.2	Wellknown URI.....	145
206	D.7.3	Resource Type	145
207	D.7.4	RAML Definition	145
208	D.7.5	Property Definition	147
209	D.7.6	CRUDN behavior.....	148
210	D.8	Ping.....	148
211	D.8.1	Introduction	148
212	D.8.2	Wellknown URI.....	148
213	D.8.3	Resource Type	148
214	D.8.4	RAML Definition	148
215	D.8.5	Property Definition	150
216	D.8.6	CRUDN behavior.....	150
217	D.9	Discoverable Resources, baseline interface	150
218	D.9.1	Introduction	150
219	D.9.2	Wellknown URI.....	150
220	D.9.3	Resource Type	150
221	D.9.4	RAML Definition	150
222	D.9.5	Property Definition	152
223	D.9.6	CRUDN behavior.....	152
224	D.10	Discoverable Resources, link list interface.....	152
225	D.10.1	Introduction	152
226	D.10.2	Wellknown URI.....	152
227	D.10.3	Resource Type	152
228	D.10.4	RAML Definition	152
229	D.10.5	Property Definition	153
230	D.10.6	CRUDN behavior.....	155
231	D.10.7	Referenced JSON schemas.....	155
232	D.10.8	oic.oic-link-schema.json	155
233	D.11	Scenes (Top level)	157
234	D.11.1	Introduction	157
235	D.11.2	Example URI	157
236	D.11.3	Resource Type	157
237	D.11.4	RAML Definition	157
238	D.11.5	Property Definition	159
239	D.11.6	CRUDN behavior.....	159
240	D.12	Scene Collections.....	160
241	D.12.1	Introduction	160
242	D.12.2	Example URI	160
243	D.12.3	Resource Type	160
244	D.12.4	RAML Definition	160

245	D.12.5	Property Definition	163
246	D.12.6	CRUDN behavior.....	164
247	D.13	Scene Member	164
248	D.13.1	Introduction	164
249	D.13.2	Example URI	164
250	D.13.3	Resource Type	164
251	D.13.4	RAML Definition	164
252	D.13.5	Property Definition	166
253	D.13.6	CRUDN behavior.....	166
254	D.14	Resource directory resource.....	166
255	D.14.1	Introduction	166
256	D.14.2	Wellknown URI.....	167
257	D.14.3	Resource Type	167
258	D.14.4	RAML Definition	167
259	D.14.5	Property Definition	171
260	D.14.6	CRUDN behavior.....	171
261	D.15	Icon	171
262	D.15.1	Introduction	171
263	D.15.2	Example URI	171
264	D.15.3	Resource Type	171
265	D.15.4	RAML Definition	172
266	D.15.5	Property Definition	173
267	D.15.6	CRUDN behavior.....	173
268	D.16	Introspection Resource.....	173
269	D.16.1	Introduction	173
270	D.16.2	Example URI	173
271	D.16.3	Resource Type	173
272	D.16.4	RAML Definition	173
273	D.16.5	Property Definition	175
274	D.16.6	CRUDN behavior.....	175
275			
276			

277
278
279

Figures

280	Figure 1: Architecture - concepts	23
281	Figure 2: Functional block diagram	24
282	Figure 3: Communication layering model	25
283	Figure 4: Example illustrating the Roles.....	26
284	Figure 5: Framework - Architecture Detail.....	26
285	Figure 6: Server bridging to Non- OCF device	27
286	Figure 7: Example of a Resource.....	30
287	Figure 8: Example - "Heater" Resource (for illustration only)	46
288	Figure 9: Example - Actuator Interface	47
289	Figure 10: Example of a Link	49
290	Figure 11: Example of distinct Links	49
291	Figure 12: Example of use of anchor in Link	50
292	Figure 13: Example of "eps Parameter	53
293	Figure 14: List of Links in a Resource.....	54
294	Figure 15: Example showing Collection and Links	55
295	Figure 16. CREATE operation	59
296	Figure 17. RETRIEVE operation	60
297	Figure 18. UPDATE operation	61
298	Figure 19. DELETE operation	62
299	Figure 20. High Level Network & Connectivity Architecture.....	64
300	Figure 21: Example of "ep"	66
301	Figure 22: Example of Link with "eps" Parameter.....	67
302	Figure 23: Example of /oic/res with Endpoint information.....	69
303	Figure 24. Resource based discovery: Information publication process.....	74
304	Figure 25. Resource based discovery: Finding information	75
305	Figure 26. Indirect discovery of resource by resource directory	84
306	Figure 27. RD discovery and RD supported query of resources support.....	85
307	Figure 28. Resource Direction Deployment Scenarios	86
308	Figure 29. Observe Mechanism	93
309	Figure 30 Generic scene resource structure	96
310	Figure 31 Interactions to check Scene support and setup of specific scenes	97
311	Figure 32 Client interactions on a specific scene	98
312	Figure 33 Interaction overview due to a Scene change	99
313	Figure 34 Interactions to check Introspection support and download the Introspection	
314	Device Data.....	103
315	Figure 35 Content-Format Policy	108
316	Figure 36. When at home: from smartphone turn on a single light.....	113

317 Figure 37. Device management (maintenance) 114
318 Figure 38. Direct interaction between Server and Client 115
319 Figure 39. Interaction between Client and Server using another Server 115
320 Figure 40. Interaction between Client and Server using Intermediary..... 115
321 Figure 41. Interaction between Client and Server using support from multiple Servers and
322 Intermediary 116
323 Figure 42. Example of Devices 116
324
325
326

DRAFT

Tables

327
328

329	Table 1. Additional OCF Types	20
330	Table 2. Name Property Definition	33
331	Table 3. Resource Identity Property Definition	33
332	Table 4. Resource Type Common Property definition	34
333	Table 5. Example foobar Resource Type	35
334	Table 6. Example foobar properties	35
335	Table 7. Resource Interface Property definition	37
336	Table 8. OCF standard Interfaces	37
337	Table 9. Common Properties for Collections (in addition to Common Properties defined in 338 section 7.3.2)	56
339	Table 10. 3rd party defined Resource elements	57
340	Table 11. Parameters of CRUDN messages	58
341	Table 12. “ep” value for Transport Protocol Suites	66
342	Table 13. List of Core Resources	70
343	Table 14. Configuration Resource	70
344	Table 15. oic.wk.con Resource Type definition	71
345	Table 16. oic.wk.con.p Resource Type definition	72
346	Table 17. Mandatory discovery Core Resources	76
347	Table 18. oic.wk.res Resource Type definition	76
348	Table 19. Protocol scheme registry.....	77
349	Table 20. oic.wk.d Resource Type definition	78
350	Table 21. oic.wk.p Resource Type definition	79
351	Table 22: Selection parameters	87
352	Table 23. Optional diagnostics and maintenance device management Core Resources	94
353	Table 24. oic.wk.mnt Resource Type definition	95
354	Table 25 list of Resource Types for Scenes	99
355	Table 26. Optional Icon Core Resource	100
356	Table 27. <i>oic.r.icon</i> Resource Type definition	100
357	Table 28. Introspection Resource	102
358	Table 29. oic.wk.introspection Resource Type definition.....	102
359	Table 30. CoAP request and response	104
360	Table 31. OCF Content-Formats.....	106
361	Table 32. Content-Format Version and Accept Version Option Numbers.....	107
362	Table 33. Accept Version and the Content-Format Version Representation	107
363	Table 34. Examples of Content-Format Version and Accept Version Representation	107
364	Table 35. Ping resource	109
365	Table 36. oic.wk.ping Resource Type definition	110
366	Table 37. oic.example.light Resource Type definition	112

367 Table 38. oic.example.garagedoor Resource Type definition 112
368 Table 39. Light control Resource Type definition 120
369 Table 40. Light control Resource Type definition 120
370 Table 41. Alphabetized list of core resources 122
371
372

DRAFT

373 1 Scope

374 The OCF specifications are divided into two sets of documents:

- 375 • Core Specification documents: The Core Specification documents specify the Framework, i.e.,
376 the OCF core architecture, interfaces, protocols and services to enable OCF profiles
377 implementation for Internet of Things (IoT) usages and ecosystems.
- 378 • Vertical Profiles Specification documents: The Vertical Profiles Specification documents
379 specify the OCF profiles to enable IoT usages for different market segments such as smart
380 home, industrial, healthcare, and automotive. The Application Profiles Specification is built
381 upon the interfaces and network security of the OCF core architecture defined in the Core
382 Specification.

383 This document is the OCF Core specification which specifies the Framework and core architecture.

384

385 2 Normative references

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

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

391 IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, August 2008

392 IETF RFC 1981, *Path MTU Discovery for IP version 6*, August 1996
393 <https://tools.ietf.org/rfc/rfc1981.txt>

394 IETF RFC 2460, *Internet Protocol, version 6 (IPv6)*, December, 1998
395 <https://tools.ietf.org/rfc/rfc2460.txt>

396 IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999.
397 <http://www.ietf.org/rfc/rfc2616.txt>

398 IETF RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*, June 2004
399 <http://www.ietf.org/rfc/rfc3810.txt>

400 IETF RFC 3986, *Uniform Resource Identifier (URI): General Syntax*, January 2005.
401 <http://www.ietf.org/rfc/rfc3986.txt>

402 IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, July 2005
403 <http://www.ietf.org/rfc/rfc4122.txt>

404 IETF RFC 4287, *The Atom Syndication Format*, December 2005,
405 <http://www.ietf.org/rfc/rfc4287.txt>

406 IETF RFC 4193, *Unique Local IPv6 Unicast Addresses*, October 2005
407 <http://www.ietf.org/rfc/rfc4193.txt>

408 IETF RFC 4291, *IP Version 6 Addressing Architecture*, February 2006
409 <http://www.ietf.org/rfc/rfc4291.txt>

410 IETF RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6*
411 *(IPv6) Specification*, March 2006
412 <http://www.ietf.org/rfc/rfc4443.txt>

413 IETF RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, September 2007
414 <http://www.ietf.org/rfc/rfc4861.txt>

415 IETF RFC 4862, *IPv6 Stateless Address Autoconfiguration*, September 2007
416 <http://www.ietf.org/rfc/rfc4862.txt>

417 IETF RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, September
418 2007
419 <http://www.ietf.org/rfc/rfc4941.txt>

420 IETF RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, September 2007
421 <http://www.ietf.org/rfc/rfc4944.txt>

422 IETF RFC 5646, *Tags for Identifying Languages*, September 2009
423 <http://www.ietf.org/rfc/rfc5646.txt>

424 IETF RFC 5988, *Web Linking: General Syntax, October 2010*
425 <http://www.ietf.org/rfc/rfc5988.txt>

426 IETF RFC 6434, *IPv6 Node Requirements*, December 2011
427 <http://www.ietf.org/rfc/rfc6434.txt>

428 IETF RFC 6455, *The WebSocket Protocol, December 2011*
429 <https://www.ietf.org/rfc/rfc6455.txt>

430 IETF RFC 6573, *The Item and Collection Link Relations*, April 2012
431 <http://www.ietf.org/rfc/rfc6573.txt>

432 IETF RFC 6690, *Constrained RESTful Environments (CoRE) Link Format*, August 2012
433 <http://www.ietf.org/rfc/rfc6690.txt>

434 IETF RFC 6762, *Multicast DNS* February 2013
435 <http://www.ietf.org/rfc/rfc6762.txt>

436 IETF RFC 6763, *DNS-Based Service Discovery*, February 2013
437 <http://www.ietf.org/rfc/rfc6763.txt>

438 IETF RFC 6775, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal
439 Area Networks (6LoWPANs)*, November 2012
440 <http://www.ietf.org/rfc/rfc6775.txt>

441 IETF RFC 7049, *Concise Binary Object Representation (CBOR)*, October 2013
442 <http://www.ietf.org/rfc/rfc7049.txt>

443 IETF RFC 7084, *Basic Requirements for IPv6 Customer Edge Routers*, November 2013
444 <http://www.ietf.org/rfc/rfc7084.txt>

445 IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014
446 <http://tools.ietf.org/rfc/rfc7159.txt>

447 IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014
448 <http://tools.ietf.org/rfc/rfc7252.txt>

449 IETF RFC 7301, *Transport Layer Security (TLS) Application-Layer Protocol Negotiation
450 Extension*, July 2014
451 <https://tools.ietf.org/html/rfc7301>

452 IETF RFC 7428, *Transmission of IPv6 Packets over ITU-T G.9959 Networks*, February 2015
453 <http://www.ietf.org/rfc/rfc7428.txt>

454 IETF RFC 7641, *Observing Resources in the Constrained Application Protocol*
455 *(CoAP)*, September 2015
456 <https://tools.ietf.org/html/rfc7641>

457 IETF RFC 7668, *IPv6 over BLUETOOTH(r) Low Energy*, October 2015
458 <https://tools.ietf.org/html/rfc7668>

459 IETF RFC 7721, *Security and Privacy Considerations for IPv6 Address Generation Mechanisms*,
460 March 2016
461 <https://tools.ietf.org/html/rfc7721>

462 IETF RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*, August
463 2016
464 <https://tools.ietf.org/html/rfc7959>

465 IETF draft-ietf-core-coap-tcp-tls-07, *CoAP over TCP, TLS, and WebSockets*, June 10 2015
466 <https://datatracker.ietf.org/doc/draft-ietf-core-coap-tcp-tls/>

467 ECMA-4-4, *The JSON Data Interchange Format*, October 2013.
468 <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>

469 OCF Security, *Open Connectivity Foundation Security Capabilities*, Version 1.0,

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

472 IANA Media Types Assignment, March 2017
473 <http://www.iana.org/assignments/media-types/media-types.xhtml>

474

475

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

478 W3C XML character escaping, *Extensible Markup Language (XML) 1.0*, November 2008
479 <http://www.w3.org/TR/2008/REC-xml-20081126/#syntax>

480 **3 Terms, definitions, symbols and abbreviations**

481 **3.1 Terms and definitions**

482 **3.1.1**

483 **Client**

484 a logical entity that accesses a Resource on a Server

485 **3.1.2**

486 **Collection**

487 a Resource that contains zero or more Links

488 **3.1.3**

489 **Configuration Source**

490 **a cloud or service network or a local read-only file which contains and provides**
491 **configuration related information to the Devices**

492 **3.1.4**

493 **Core Resources**

494 those Resources that are defined in this specification

495 **3.1.5**

496 **Default Interface**

497 an Interface used to generate the response when an Interface is omitted in a request

498 **3.1.6**

499 **Device**

500 a logical entity that assumes one or more Roles (e.g., Client, Server)

501 Note 1 to entry: More than one Device can exist on a physical platform.

502 **3.1.7**

503 **Device Type**

504 a uniquely named definition indicating a minimum set of Resource Types that a Device supports

505 Note 1 to entry: A Device Type provides a hint about what the Device is, such as a light or a fan, for use during
506 Resource discovery.

507 **3.1.8**

508 **Entity**

509 **an aspect of the physical world that is exposed through a Device**

510 Note 1 to entry: Example of an entity is an LED.

511 **3.1.9**

512 **Framework**

513 a set of related functionalities and interactions defined in this specification, which enable
514 interoperability across a wide range of networked devices, including IoT

515 **3.1.10**

516 **Interface**

517 provides a view and permissible responses on a Resource

518 **3.1.11**
519 **Introspection**
520 mechanism to determine the capabilities of the hosted Resources of a Device

521 **3.1.12**
522 **Introspection Device Data**
523 data that describes the payloads per implemented method of the Resources that makes up the
524 Device

525 Note 1 to entry: See section 11.8 for all requirements and exceptions

526 **3.1.13**
527 **Links**
528 extends typed web links according to IETF RFC 5988

529 **3.1.14**
530 **Non-OCF Device**
531 A device which does not comply with the OCF Device requirements

532 **3.1.15**
533 **Notification**
534 the mechanism to make a Client aware of resource state changes in a Resource

535 **3.1.16**
536 **Observe**
537 the act of monitoring a Resource by sending a RETRIEVE request which is cached by the Server
538 hosting the Resource and reprocessed on every change to that Resource

539 **3.1.17**
540 **Parameter**
541 an element that provides metadata about a Resource referenced by the target URI of a Link

542 **3.1.18**
543 **Partial UPDATE**
544 an UPDATE request to a Resource that includes a subset of the Properties that are visible via the
545 Interface being applied for the Resource Type

546 **3.1.19**
547 **Platform**
548 a physical device containing one or more Devices

549 **3.1.20**
550 **Remote Access Endpoint (RAE) Client**
551 a Client which supports XMPP functionality in order to access a Server from a remote location

552 **3.1.21**
553 **Remote Access Endpoint (RAE) Server**
554 a Server which supports XMPP and can publish its resource(s) to an XMPP server in the cloud,
555 thus becoming remotely addressable and accessible

556 Note 1 to entry: An RAE Server also supports ICE/STUN/TURN.

557 **3.1.22**
558 **Resource**
559 represents an Entity modelled and exposed by the Framework

560 **3.1.23**
561 **Resource Directory**
562 a set of descriptions of Resources where the actual Resources are held on Servers external to the
563 Device hosting the Resource Directory, allowing lookups to be performed for those resources

564 Note 1 to entry: This functionality can be used by sleeping Servers or Servers that choose not to listen/respond to
565 multicast requests directly.

566 **3.1.24**
567 **Resource Interface**
568 a qualification of the permitted requests on a Resource

569 **3.1.25**
570 **Resource Property**
571 a significant aspect or parameter of a resource, including metadata, that is exposed through the
572 Resource

573 **3.1.26**
574 **Resource Type**
575 a uniquely named definition of a class of Resource Properties and the interactions that are
576 supported by that class

577 Note 1 to entry: Each Resource has a Property "rt" whose value is the unique name of the Resource Type.

578 **3.1.27**
579 **Scene**
580 a static entity that stores a set of defined Resource property values for a collection of Resources

581 Note 1 to entry: A Scene is a prescribed setting of a set of resources with each having a predetermined value for the
582 property that has to change.

583 **3.1.28**
584 **Scene Collection**
585 a collection Resource that contains an enumeration of possible Scene Values and the current
586 Scene Value

587 Note 1 to entry: The member values of the Scene collection Resource are Scene Members.

588 **3.1.29**
589 **Scene Member**
590 a Resource that contains mappings of Scene Values to values of a property in the resource

591 **3.1.30**
592 **Scene Value**
593 a Scene enumerator representing the state in which a Resource can be

594 **3.1.31**
595 **Server**
596 a Device with the role of providing resource state information and facilitating remote interaction
597 with its resources

598 Note 1 to entry: A Server can be implemented to expose non-OCF Device resources to Clients (section 5.6)

599 **3.2 Symbols and abbreviations**

600 **3.2.1**
601 **ACL**
602 Access Control List

603 Note 1 to entry: The details are defined in OCF Security.

604 **3.2.2**
605 **CBOR**
606 Concise Binary Object Representation

607 **3.2.3**
608 **CoAP**
609 Constrained Application Protocol

610 **3.2.4**
611 **EXI**
612 Efficient XML Interchange

613 **3.2.5**
614 **IRI**
615 Internationalized Resource Identifiers

616 **3.2.6**
617 **ISP**
618 Internet Service Provider

619 **3.2.7**
620 **JSON**
621 JavaScript Object Notation

622 **3.2.8**
623 **mDNS**
624 Multicast Domain Name Service

625 **3.2.9**
626 **MTU**
627 Maximum Transmission Unit

628 **3.2.10**
629 **NAT**
630 Network Address Translation

631 **3.2.11**
632 **OCF**
633 Open Connectivity Foundation
634 the organization that created this specification

635 **3.2.12**
636 **RAML**
637 RESTful API Modeling Language

638 **3.2.13**
639 **REST**
640 Representational State Transfer

641 **3.2.14**
642 **RESTfull**
643 REST-compliant Web services

644 **3.2.15**
645 **URI**
646 Uniform Resource Identifier

647 **3.2.16**
648 **URN**
649 Uniform Resource Name

650 **3.2.17**
651 **UTC**
652 Coordinated Universal Time

653 **3.2.18**
654 **UUID**
655 Universal Unique Identifier

656 **3.2.19**
657 **XML**
658 Extensible Markup Language

659 **3.3 Conventions**

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

664 **3.4 Data types**

665 Resources are defined using data types derived from JSON values as defined in ECMA-4-4.
666 However, a Resource can overload a JSON defined value to specify a particular subset of the
667 JSON value, using validation keywords defined in [JSON Schema Validation].

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

675

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

677

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])\$	As defined in ISO 8601. The format is [yyyy]-[mm]-[dd].
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 section 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})^*}$$	An IETF language tag formatted according to IETF RFC 5646 section 2.1.
uint64	$^{0 ([1-9][0-9]{0,19})}$$	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 section 3.

678

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

680

681 In a JSON schema, “maxLength” for a string indicates the maximum number of characters not
682 octets. However, “maxLength” shall also indicate the maximum number of octets. If no “maxLength”
683 is defined for a string, then the maximum length shall be 64 octets.

684 4 Document conventions and organization

685 In this document, features are described as required, recommended, allowed or DEPRECATED as
686 follows:

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

- 688 • These basic features shall be implemented to comply with Core Architecture. The phrases
689 “shall not”, and “PROHIBITED” indicate behaviour that is prohibited, i.e. that if performed
690 means the implementation is not in compliance.

691 Recommended (or should)(S).

- 692 • These features add functionality supported by Core Architecture and should be implemented.
693 Recommended features take advantage of the capabilities Core Architecture, usually without
694 imposing major increase of complexity. Notice that for compliance testing, if a recommended
695 feature is implemented, it shall meet the specified requirements to be in compliance with these
696 guidelines. Some recommended features could become requirements in the future. The phrase
697 “should not” indicates behaviour that is permitted but not recommended.

698 Allowed (may or allowed)(O).

- 699 • These features are neither required nor recommended by Core Architecture, but if the feature
700 is implemented, it shall meet the specified requirements to be in compliance with these
701 guidelines.

702 DEPRECATED.

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

709 Conditionally allowed (CA)

710 • The definition or behaviour depends on a condition. If the specified condition is met, then the
711 definition or behaviour is allowed, otherwise it is not allowed.

712 Conditionally required (CR)

713 • The definition or behaviour depends on a condition. If the specified condition is met, then the
714 definition or behaviour is required. Otherwise the definition or behaviour is allowed as default
715 unless specifically defined as not allowed.

716

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

718 Words that are emphasized are printed in italic.

719 **5 Architecture**

720 **5.1 Overview**

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

725 Specifically, the architecture provides:

- 726 • A communication and interoperability framework for multiple market segments (Consumer,
727 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,
728 transports and use cases
- 729 • A common and consistent model for describing the environment and enabling information
730 and semantic interoperability
- 731 • Common communication protocols for discovery and connectivity
- 732 • Common security and identification mechanisms
- 733 • Opportunity for innovation and product differentiation
- 734 • A scalable solution addressing different device capabilities, applicable to smart devices as
735 well as the smallest connected things and wearable devices

736 The architecture is based on the Resource Oriented Architecture design principles and described
737 in the sections 5.2 through 5.6 respectively. Section 5.2 presents the guiding principles for OCF
738 operations. Section 5.3 defines the functional block diagram and Framework. Section 5.5 provides
739 an example scenario with roles. Section 5.6 provides an example scenario of bridging to non- OCF
740 ecosystem.

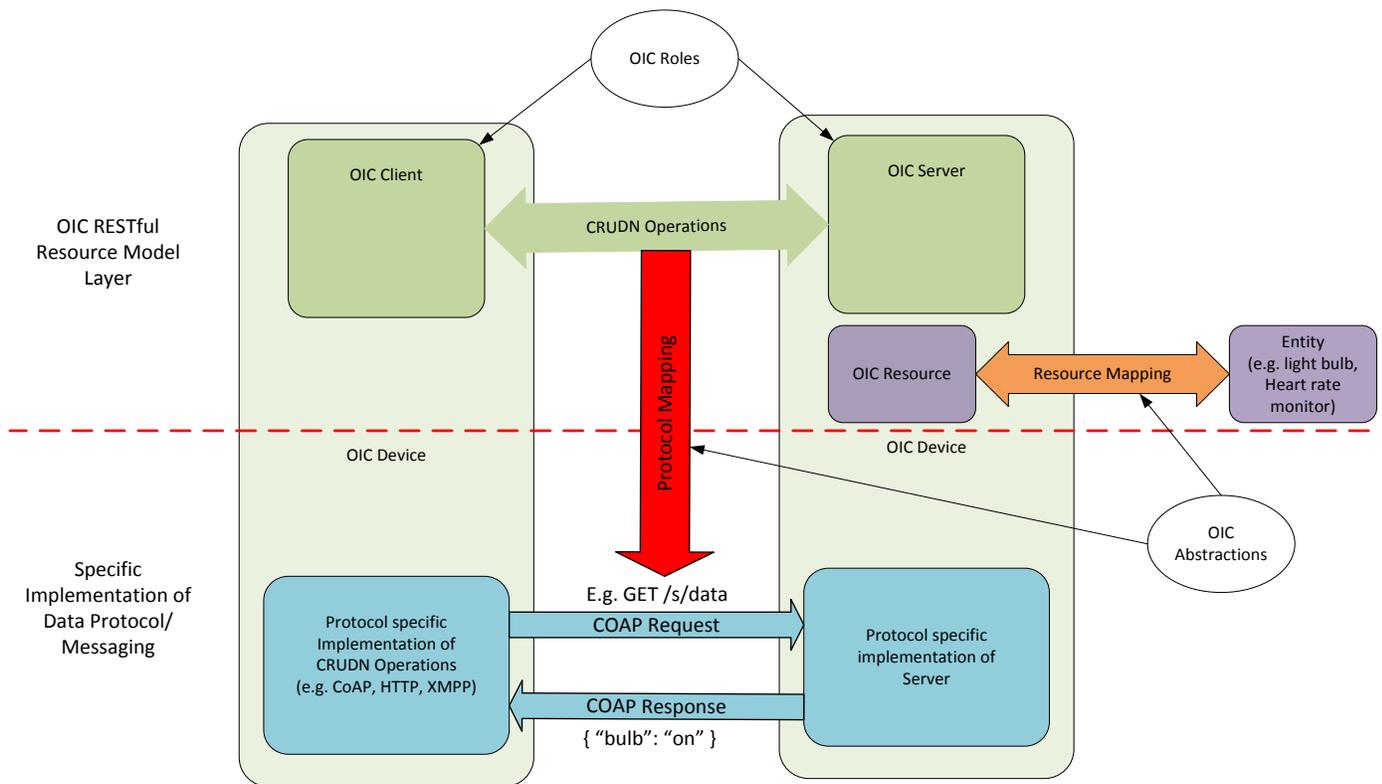
741 **5.2 Principle**

742 In the architecture, Entities in the physical world (e.g., temperature sensor, an electric light or a
743 home appliance) are represented as resources. Interactions with an Entity are achieved through
744 its resource representations (section 7.7) using operations that adhere to Representational State
745 Transfer (REST) architectural style, i.e., RESTful interactions.

746 The architecture defines the overall structure of the Framework as an information system and the
747 interrelationships of the Entities that make up OCF. Entities are exposed as Resources, with their
748 unique identifiers (URIs) and support interfaces that enable RESTful operations on the Resources.
749 Every RESTful operation has an initiator of the operation (the client) and a responder to the

750 operation (the server). In the Framework, the notion of the client and server is realized through
 751 roles (section 5.5). Any Device can act as a Client and initiate a RESTful operation on any Device
 752 acting as a Server. Likewise, any Device that exposes Entities as Resources acts as a Server.
 753 Conformant to the REST architectural style, each RESTful operation contains all the information
 754 necessary to understand the context of the interaction and is driven using a small set of generic
 755 operations, i.e., CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY (CRUDN) defined in
 756 section 8, which include representations of Resources.

757 Figure 1 depicts the architecture.



758
759

760 **Figure 1: Architecture - concepts**

761

762 The architecture is organized conceptually into three major aspects that provide overall separation
 763 of concern: resource model, RESTful operations and abstractions.

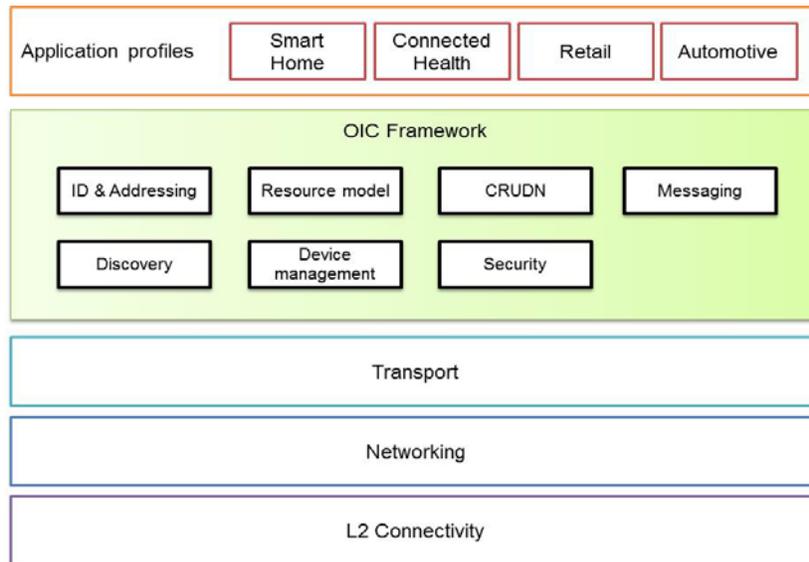
- 764
- 765 • Resource model: The resource model provides the abstractions and concepts required to
 766 logically model, and logically operate on the application and its environment. The core resource
 767 model is common and agnostic to any specific application domain such as smart home,
 768 industrial or automotive. For example, the resource model defines a Resource which abstracts
 769 an Entity and the representation of a Resource maps the Entity's state. Other resource model
 concepts can be used to model other aspects, for example behaviour.
 - 770 • RESTful operations: The generic CRUDN operations are defined using the RESTful paradigm
 771 to model the interactions with a Resource in a protocol and technology agnostic way. The
 772 specific communication or messaging protocols are part of the protocol abstraction and
 773 mapping of Resources to specific protocols is provided in section 11.8.

774 • Abstraction: The abstractions in the resource model and the RESTful operations are mapped
 775 to concrete elements using abstraction primitives. An entity handler is used to map an Entity
 776 to a Resource and connectivity abstraction primitives are used to map logical RESTful
 777 operations to data connectivity protocols or technologies. Entity handlers may also be used to
 778 map Resources to Entities that are reached over protocols that are not natively supported by
 779 OCF.

780

781 5.3 Functional block diagram

782 The functional block diagram encompasses all the functionalities required for operation. These
 783 functionalities are categorized as L2 connectivity, networking, transport, Framework, and
 784 application profiles. The functional blocks are depicted in Figure 2 and listed below.

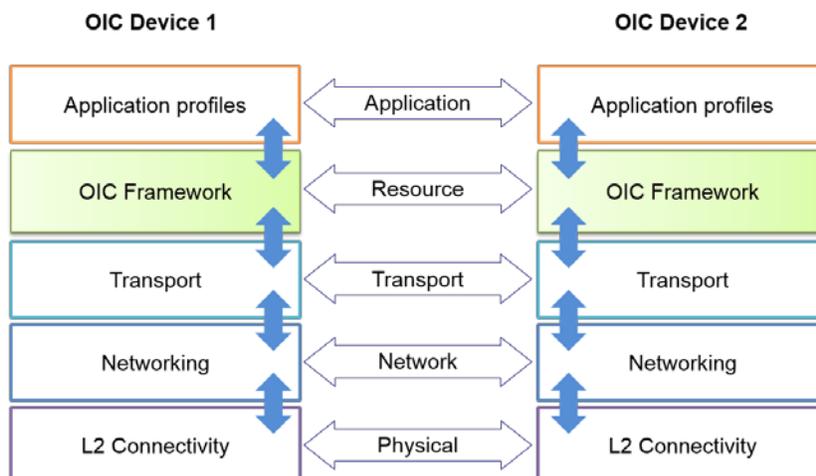


785

786 **Figure 2: Functional block diagram**

- 787 • **L2 connectivity:** Provides the functionalities required for establishing physical and data
 788 link layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.
- 789 • **Networking:** Provides functionalities required for Devices to exchange data among
 790 themselves over the network (e.g., Internet).
- 791 • **Transport:** Provides end-to-end flow transport with specific QoS constraints. Examples of
 792 a transport protocol include TCP and UDP or new Transport protocols under development
 793 in the IETF, e.g., Delay Tolerant Networking (DTN).
- 794 • **Framework:** Provides the core functionalities as defined in this specification. The
 795 functional block is the source of requests and responses that are the content of the
 796 communication between two Devices.
- 797 • **Application profile:** Provides market segment specific data model and functionalities, e.g.,
 798 smart home data model and functions for the smart home market segment.

799 When two Devices communicate with each other, each functional block in a Device interacts with
 800 its counterpart in the peer Device as shown in Figure 3.



801
802 **Figure 3: Communication layering model**

803 **5.4 Framework**

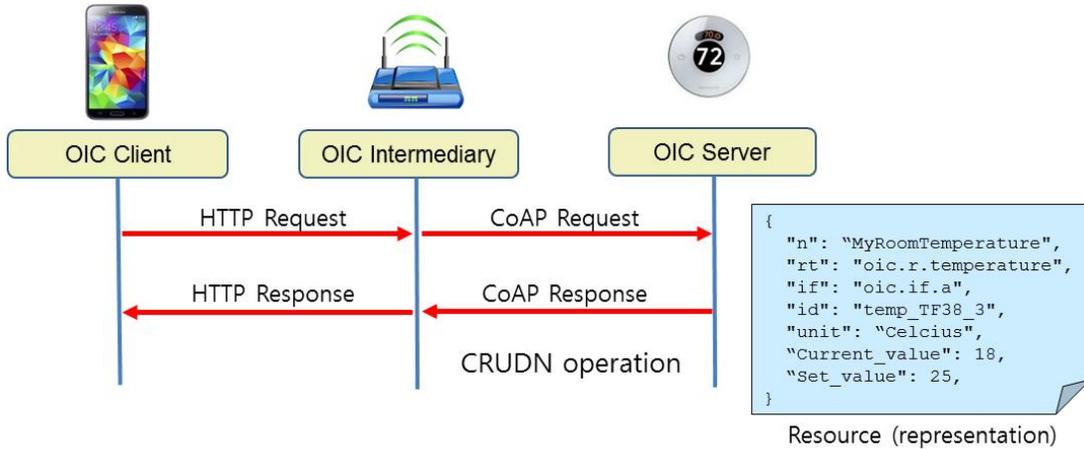
804 Framework consists of functions which provide core functionalities for operation.

- 805 1) **Identification and addressing.** Defines the identifier and addressing capability. The
806 Identification and addressing function is defined in section 6.
- 807 2) **Discovery.** Defines the process for discovering available
808 a) Devices (Endpoint Discovery in section 10) and
809 b) Resources (Resource discovery in section 11.3)
- 810 3) **Resource model.** Specifies the capability for representation of Entities in terms of resources
811 and defines mechanisms for manipulating the resources. The resource model function is
812 defined in section 7.
- 813 4) **CRUDN.** Provides a generic scheme for the interactions between a Client and Server as
814 defined in section 8.
- 815 5) **Messaging.** Provides specific message protocols for RESTful operation, i.e. CRUDN. For
816 example, CoAP is a primary messaging protocol. The messaging function is defined in section
817 11.8.
- 818 6) **Device management.** Specifies the discipline of managing the capabilities of a Device, and
819 includes device provisioning and initial setup as well as device monitoring and diagnostics.
820 The device management function is defined in section 11.5.
- 821 7) **Security.** Includes authentication, authorization, and access control mechanisms required for
822 secure access to Entities. The security function is defined in section 13.

823 **5.5 Example Scenario with roles**

824 Interactions are defined between logical entities known as Roles. Three roles are defined: Client,
825 Server and Intermediary.

826 Figure 4 illustrates an example of the Roles in a scenario where a smart phone sends a request
827 message to a thermostat; the original request is sent over HTTP, but is translated into a CoAP
828 request message by a gateway in between, and then delivered to the thermostat. In this example,
829 the smart phone takes the role of a Client, the gateway takes the role of an Intermediary and the
830 thermostat takes the role of a Server.



831

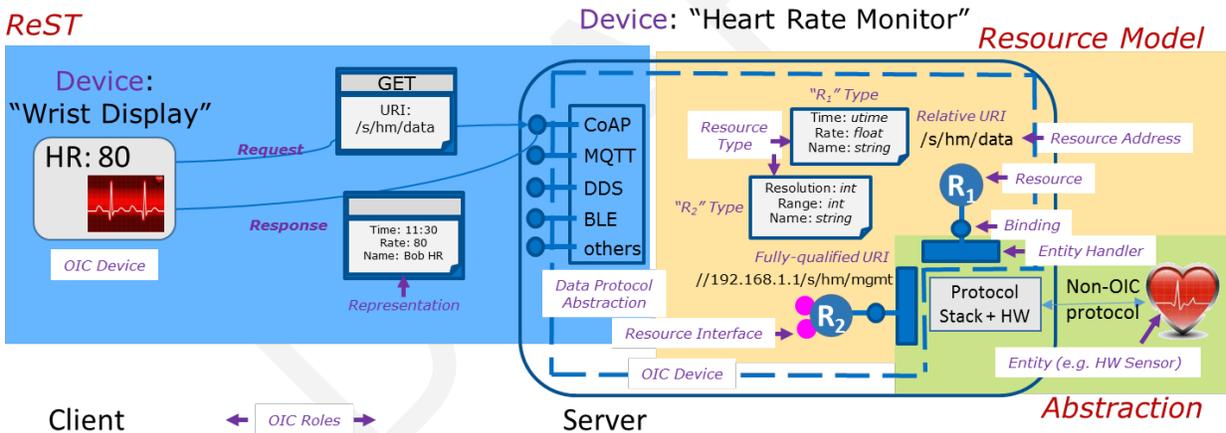
832

Figure 4: Example illustrating the Roles

833 **5.6 Example Scenario: Bridging to Non- OCF ecosystem**

834 The use case for this scenario is a display (like a wrist watch) that is used to monitor a heart rate
 835 sensor that implements a protocol that is not OCF supported.

836 Figure 5 provides a detailed logical view of the concepts described in Figure 1.



837

838

Figure 5: Framework - Architecture Detail

839

840 The details may be implemented in many ways, for example, by using a Server with an entity
 841 handler to interface directly to a non- OCF device as shown in Figure 6.



842
843

844

Figure 6: Server bridging to Non- OCF device

845 On start-up the Server runs the entity handlers which discover the non- OCF systems (e.g., Heart
846 Rate Sensor Device) and create resources for each device or functionality discovered. The entity
847 handler creates a Resource for each discovered device or functionality and binds itself to that
848 Resource. These resources are made discoverable by the Server.

849 Once the resources are created and made discoverable, then the Display Device can discover
850 these resources and operate on them using the mechanisms described in this specification. The
851 requests to a resource on the Server are then interpreted by the entity handler and forwarded to
852 the non- OCF device using the protocol supported by the non-OCF device. The returned
853 information from the non- OCF device is then mapped to the appropriate response for that resource.

854 **6 Identification and addressing**

855 **6.1 Introduction**

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

858 The *identifier* shall unambiguously and uniquely identify an element in a context or domain. The
859 context or domain may be determined by the use or the application. The identifier should be
860 immutable over the lifecycle of that element and shall be unique within a context or domain.

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

863 The *name* is a handle that distinguishes the element from other elements in the framework. The
864 name may be changed over the lifecycle of that element.

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

867 Each of these aspects may be defined separately for multiple contexts (e.g., a context could be a
868 layer in a stack). So an address may be a URL for addressing resource and an IP address for
869 addressing at the connectivity layer. In some situations, both these addresses would be required.
870 For example, to do RETRIEVE (section 8.3) operation on a particular resource representation, the
871 client needs to know the address of the target resource and the address of the server through
872 which the resource is exposed.

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

875 The remainder of this section discusses the identifier, address and naming from the point of view
876 of the resource model and the interactions to be supported by the resource model. Examples of
877 interactions are the RESTful interactions, i.e. CRUDN operation (section 8) on a resource. Also
878 the mapping of these to transport protocols, e.g., CoAP is described.

879 6.2 Identification

880 An identifier shall be unique within the context or domain of use. There are many schemes that
881 may be used to generate an identifier that has the required properties. The identifier may be
882 context-specific in that the identifier is expected to be and guaranteed to be unique only within that
883 context or domain. Identifier may also be context-independent where these identifiers are
884 guaranteed to be unique across all contexts and domains both spatially and temporally. The
885 context-specific identifiers could be defined by simple schemes like monotonic enumeration or may
886 be defined by overloading an address or name, for example an IP address may be an identifier
887 within the private domain behind a gateway in a smart home. On the other hand, context-
888 independent identifiers require a stronger scheme that derives universally unique identities, for
889 example any one of the versions of Universally Unique Identifiers (UUIDs). Context independent
890 identifier may also be generated using hierarchy of domains where the root of the hierarchy is
891 identified with a UUID and sub-domains may generate context independent identifier by
892 concatenating context-specific identifiers for that domain to the context-independent identifier of
893 their parent.

894 6.2.1 Resource identification and addressing

895 A resource may be identified using a URI and addressed by the same URI if the URI is a URL. In
896 some cases a resource may need an identifier that is different from a URI; in this case, the resource
897 may have a property whose value is the identifier. When the URI is in the form of a URL, then the
898 URI may be used to address the resource.

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

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

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

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

903 A description of values that each component takes is given below.

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

907 Each transport binding is responsible for specifying how an OCF URI is converted to a transport
908 protocol URI before sending over the network by the requestor. Similarly on the receiver side, each
909 transport binding is responsible for specifying how an OCF URI is converted from a transport
910 protocol URI before handing over to the resource model layer on the receiver.

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

913 The *path* shall be a unique string that unambiguously identifies or references a resource within the
914 context of the Server. In this version of the specification, a path shall not include pct-encoded non-
915 ASCII characters or NUL characters. A *path* shall be preceded by a '/' (slash). The *path* may have
916 '/' (slash) separated segments for human readability reasons. In the OCF context, the '/' (slash)
917 separated segments are treated as a single string that directly references the resources (i.e. a flat
918 structure) and not parsed as a hierarchy. On the Server, the path or some substring in the path
919 may be shortened by using hashing or some other scheme provided the resulting reference is
920 unique within the context of the host.

921 Once a path is generated, a client accessing the resource or recipient of the URI shall use that
922 path as an opaque string and shall NOT parse to infer a structure, organization or semantic.

923 A query string shall contain a list of <name>=<value> segments (aka “name-value pair”) each
924 separated by a ‘&’ (ampersand). The query string will be mapped to the appropriate syntax of the
925 protocol used for messaging. (e.g., CoAP).

926 A URI may be either

- 927 • Fully qualified or
- 928 • Relative

929 *Generation of URI:*

930 A URI may be defined by the Client which is the creator of that resource. Such a URI may be
931 relative or absolute (fully qualified). A relative URI shall be relative to the Device on which it is
932 hosted. Alternatively, a URI may be generated by the Server of that resource automatically based
933 on a pre-defined convention or organization of the resources, based on an interface, based on
934 some rules or with respect to different roots or bases.

935 *Use of URI:*

936 The absolute path reference of a URI is to be treated as an opaque string and a client shall not
937 infer any explicit or implied structure in the URI – the URI is simply an address. It is also
938 recommended that Devices hosting a resource treat the URI of each resource as an opaque string
939 that addresses only that resource. (e.g., URI's /a and /a/b are considered as distinct addresses
940 and resource b cannot be construed as a child of resource a).

941 **6.3 Namespace:**

942 The relative URI prefix “/oic/” is reserved as a namespace for URIs defined in OCF specifications
943 and shall not be used for URIs that are not defined in OCF specifications.

944 **6.4 Network addressing**

945 The following are the addresses used in this specification:

- 946 • **IP address**

947 An IP address is used when the device is using an IP configured interface.

948 When a Device only has the identity information of its peer, a resolution mechanism is needed to
949 map the identifier to the corresponding address.

950 **7 Resource model**

951 **7.1 Introduction**

952 The Resource Model defines concepts and mechanisms that provide consistency and core
953 interoperability between devices in the OCF ecosystems. The Resource Model concepts and
954 mechanisms are then mapped to the transport protocols to enable communication between the
955 devices – each transport provides the communication protocol interoperability. The Resource
956 Model, therefore, allows for interoperability to be defined independent of the transports.

957 In addition, the concepts in the Resource Model support modelling of the primary artefacts and
958 their relationships to one and another and capture the semantic information required for
959 interoperability in a context. In this way, OCF goes beyond simple protocol interoperability to
960 capture the rich semantics required for true interoperability in Wearable and Internet of Things
961 ecosystems.

962 The primary concepts in the Resource Model are: Entity, Resources, Uniform Resource Identifiers
963 (URI), Resource Types, Properties, Representations, Interfaces, Collections and Links. In addition,
964 the general mechanisms are CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY. These
965 concepts and mechanisms may be composed in various ways to define the rich semantics and
966 interoperability needed for a diverse set of use cases that the OCF framework is applied to.

967 In the OCF Resource Model framework, an Entity needs to be visible, interacted with or
968 manipulated, it is represented by an abstraction called a Resource. A Resource encapsulates and
969 represents the state of an Entity. A Resource is identified, addressed and named using URIs.

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

974 A resource instance is derived from a Resource Type. The uni-directional relationship between
975 one Resource and another Resource is defined as a Link. A Resource that has Properties and
976 Links is a Collection.

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

980 A Resource (and Resource Type) could represent and be used to expose a capability. Interactions
981 with that Resource can be used to exercise or use that capability. Such capabilities can be used
982 to define processes like discovery, management, advertisement etc. For example: "discovery of
983 resources on a device" can be defined as the retrieval of a representation of a specific resource
984 where a property or properties have values that describe or reference the resources on the device.

985 The information for Request or Response with the Representation may be communicated "on the
986 wire" by serializing using a transfer protocol or encapsulated in the payload of the transport
987 protocol – the specific method is determined by the normative mapping of the Request or Response
988 to the transport protocol. See section 11.8 for transport protocols supported.

989 The RAML definitions used in this document are normative. This also includes that all defined
990 JSON payloads shall comply with the indicated JSON schema. See Annex D for Resource Types
991 defined in this specification.

992 **7.2 Resource**

993 A Resource shall be defined by one or more Resource Type(s) – see Annex D for Resource Type.
994 A request to CREATE a Resource shall specify one or more Resource Types that define that
995 Resource.

996 A Resource is hosted in a Device. A Resource shall have a URI as defined in section 6. The URI
997 may be assigned by the Authority at the creation of the Resource or may be pre-defined by the
998 specification of the Resource Type.

```
999 /my/resource/example } URI  
1000 {  
  "rt": "oic.r.foobar",  
  "if": "oic.if.a",  
  "value": "foo value"  
} } Properties
```

Figure 7: Example of a Resource

1001

1002 Core Resources are the Resources defined in this specification to enable functional interactions
1003 as defined in section 10 (e.g., Discovery, Device Management, etc). Among the Core Resources,
1004 /oic/res, /oic/p, and oic/d shall be supported on all Devices. Devices may support other Core
1005 Resources depending on the functional interactions they support.

1006 7.3 Property

1007 7.3.1 Introduction

1008 A Property describes an aspect that is exposed through a Resource including meta-information
1009 related to that resource.

1010 A Property shall have a name i.e. Property Name and a value i.e. Property Value. The Property is
1011 expressed as a key-value pair where key is the Property Name and value the Property Value like
1012 <Property Name> = <Property Value>. For example if the “temperature” Property has a Property
1013 Name “temp” and a Property Value “30F”, then the Property is expressed as “temp=30F”. The
1014 specific format of the Property depends on the encoding scheme. For example, in JSON, Property
1015 is represented as "key": value (e.g., "temp": 30).

1016 In addition, the Property definition shall have a

- 1017 • **Value Type** – the Value Type defines the values that a Property Value may take. The Value
1018 Type may be a simple data type (e.g. string, Boolean) as defined in section 3.4 or may be a
1019 complex data type defined with a schema. The Value Type may define
 - 1020 ○ Value Rules define the rules for the set of values that the Property Value may take.
1021 Such rules may define the range of values, the min-max, formulas, set of
1022 enumerated values, patterns, conditional values and even dependencies on values
1023 of other Properties. The rules may be used to validate the specific values in a
1024 Property Value and flag errors.
- 1025 • **Mandatory** – specifies if the Property is mandatory or not for a given Resource Type.
- 1026 • **Access modes** – specifies whether the Property may be read, written or both. Updates are
1027 equivalent to a write. “r” is used for read and “w” is used for write – both may be specified.
1028 Write does not automatically imply read.

1029 The definition of a Property may include the following additional information – these items are
1030 informative:

- 1031 • **Property Title** - a human-friendly name to designate the Property; usually not sent over the
1032 wire
- 1033 • **Description** – descriptive text defining the purpose and expected use of this Property.

1034 A Property may be used in the query part of an URI as one criterion for selection of a particular
1035 Resource. This is done by declaring the Property (i.e. <Property Name> = <desired Property
1036 Value>) as one of the segments of the query. In this version of the specification, only ASCII strings
1037 are permitted in query filters, and NUL characters are disallowed in query filters. This means that
1038 only property values with ASCII characters can be matched in a query filter. The Resource is
1039 selected when all the declared Properties in the query match the corresponding Properties in the
1040 full Representation of the target Resource. The full Representation is the snapshot that includes
1041 the union of all Properties in all Resource Types that define the target Resource. If the Property is
1042 declared in the “filter” segment of the query then the declared Property is matched to the
1043 Representation defined by the Interface to isolate certain parts of that Representation.

1044 In general, a property is meaningful only within the resource to which it is associated. However a
1045 base set of properties that may be supported by all Resources, known as Common Properties,
1046 keep their semantics intact across Resources i.e. their “key=value” pair means the same in any

1047 Resource. Detailed tables with the above fields for all common properties are defined in section
1048 7.3.2.

1049 **7.3.2 Common Properties**

1050 **7.3.2.1 Introduction**

1051 The Common Properties defined in this section may be specified for all Resources. The following
1052 Properties are defined as Common Properties: “Resource Type”, “Resource Interface”, “Name”,
1053 and “Resource Identity”.

1054 The name of a Common Property shall be unique and shall not be used by other properties. When
1055 defining a new Resource Type, its non-common properties shall not use the name of existing
1056 Common Properties (e.g., “rt”, “if”, “n”, “id”). When defining a new “Common Property”, it should
1057 be ensured that its name has not been used by any other properties. The uniqueness of a new
1058 Common Property name can be verified by checking all the Properties of all the existing OCF
1059 defined Resource Types. However, this may become cumbersome as the number of Resource
1060 Types grow. To prevent such name conflicts in the future, OCF may reserve a certain name space
1061 for common property. Potential approaches are (1) a specific prefix (e.g. “oic”) may be designated
1062 and the name preceded by the prefix (e.g. “oic.psize”) is only for Common Property; (2) the names
1063 consisting of one or two letters are reserved for Common Property and all other Properties shall
1064 have the name with the length larger than the 2 letters; (3) Common Properties may be nested
1065 under specific object to distinguish themselves.

1066 The ability to UPDATE a Common Property (that supports write as an access mode) is restricted
1067 to the oic.if.rw (read-write) Interface; thus a Common Property shall be updatable using the read-
1068 write Interface if and only if the Property supports write access as defined by the Property definition
1069 and the associated schema for the read-write Interface.

1070 The following Common Properties for all Resources are specified in section 7.3.2.2 through section
1071 7.3.2.6 and summarized as follows:

- 1072 • Resource Type (“rt”) – this Property is used to declare the Resource Type of that Resource.
1073 Since a Resource could be define by more than one Resource Type the Property Value of the
1074 Resource Type Property can be used to declare more than one Resource type. For example:
1075 “rt”: [“oic.wk.d”, “oic.d.airconditioner”] declares that the Resource containing this Property is
1076 defined by either the “oic.wk.d” Resource Type or the “oic.d.airconditioner” Resource Type.
1077 See section 7.3.2.3 for details.
- 1078 • Interface (“if”) – this Property declares the Interfaces supported by the Resource. The Property
1079 Value of the Interface Property can be multi-valued and lists all the Interfaces supported. See
1080 section 7.3.2.4 for details.
- 1081 • Name (“n”) – the Property declares “human-readable” name assigned to the Resource. See
1082 section 7.3.2.5.
- 1083 • Resource Identity (“id”): its Property Value shall be a unique (across the scope of the host
1084 Server) instance identifier for a specific instance of the Resource. The encoding of this identifier
1085 is device and implementation dependent. See section 7.3.2.6 for details.

1086 **7.3.2.2 Property Name and Property Value definitions**

1087 The Property Name and Property Value as used in this specification:

- 1088 • **Property Name**– the key in “key=value” pair. Property Name is case sensitive and its data type
1089 is “string” but only ASCII characters are permitted, and embedded NUL characters are not
1090 permitted.
- 1091 • **Property Value** – the value in “key=value” pair. Property Value is case sensitive when its data
1092 type is “string”. Any enum values shall be ASCII only.

1093 **7.3.2.3 Resource Type**

1094 Resource Type Property is specified in section 7.4.

1095 **7.3.2.4 Interface**

1096 Interface Property is specified in Section 7.5.

1097 **7.3.2.5 Name**

1098 A human friendly name for the Resource, i.e. a specific resource instance name (e.g.,
1099 MyLivingRoomLight), The Name Property is as defined in Table 2

1100 **Table 2. Name Property Definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	Human understandable name for the resource.

1101 The 'Name' Property is read-write unless otherwise restricted by the Resource Type (i.e. the
1102 Resource Type does not support UPDATE or does not support UPDATE using read-write).

1103 **7.3.2.6 Resource Identity**

1104 The Resource Identity Property shall be a unique (across the scope of the host Server) instance
1105 identifier for a specific instance of the Resource. The encoding of this identifier is device and
1106 implementation dependent. The Resource Identity Property is as defined in Table 3.

1107 **Table 3. Resource Identity Property Definition**

1108

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

1109

1110 **7.4 Resource Type**

1111 **7.4.1 Introduction**

1112 Resource Type is a class or category of Resources and a Resource is an instance of one or more
1113 Resource Types.

1114 The Resource Types of a Resource is declared using the Resource Type Common Property as
1115 described in Section 7.3.2.3 or in a Link using the Resource Type Parameter.

1116 A Resource Type may either be pre-defined (Core Resource Types in this specification and vertical
1117 Resource Types in vertical domain specifications) or in custom definitions by manufacturers, end
1118 users, or developers of Devices (vendor-defined Resource Types). Resource Types and their
1119 definition details may be communicated out of band (i.e. in documentation) or be defined explicitly
1120 using a meta-language which may be downloaded and used by APIs or applications. OCF has
1121 adopted RAML and JSON Schema as the specification method for OCF's RESTful interfaces and
1122 Resource definitions.

1123 Every Resource Type shall be identified with a Resource Type ID which shall be represented using
1124 the requirements and ABNF governing the Resource Type attribute in IETF RFC 6690(Section 2

1125 for ABNF and Section 3.1 for requirements) with the caveat that segments are separated by a "."
 1126 (period). The entire string represents the Resource Type ID. When defining the ID each segment
 1127 may represent any semantics that are appropriate to the Resource Type. For example, each
 1128 segment could represent a namespace. Once the ID has been defined, the ID should be used
 1129 opaquely and an implementations should not infer any information from the individual segments.
 1130 The string "oic", when used as the first segment in the definition of the Resource Type ID, is
 1131 reserved for OCF-defined Resource Types. All OCF defined Resource Types are to be registered
 1132 with the IANA Core Parameters registry as described also in IETF RFC 6690.

1133 7.4.2 Resource Type Property

1134 A Resource when instantiated or created shall have one or more Resource Types that are the
 1135 template for that Resource. The Resource Types that the Resource conforms to shall be declared
 1136 using the "rt" Common Property for the Resource. The Property Value for the "rt" Common Property
 1137 shall be the list of Resource Type IDs for the Resource Types used as templates (i.e., "rt"=<list of
 1138 Resource Type IDs>).

1139 **Table 4. Resource Type Common Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource type	rt	json	Array of Resource Type IDs		R	yes	The property name rt is as described in IETF RFC 6690

1140 Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and
 1141 the host (i.e. Server) of the Resource.

1142 7.4.3 Resource Type definition

1143 Resource Type is specified as follows:

- 1144 • **Pre-defined URI (optional)** – a pre-defined URI may be specified for a specific Resource Type
 1145 in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that
 1146 Resource Type shall use only the pre-defined URI. An instance of a different Resource Type
 1147 shall not use the pre-defined URI.
- 1148 • **Resource Type Title (optional)** – a human friendly name to designate the Resource Type.
- 1149 • **Resource Type ID** – the value of "rt" property which identifies the Resource Type, (e.g.,
 1150 oic.wk.p).
 - 1151 • **Resource Interfaces** – list of the interfaces that may be supported by the Resource Type.
 - 1152 • **Resource Properties** – definition of all the properties that apply to the Resource Type. The
 1153 Resource Type definition shall define whether a property is mandatory, conditional mandatory,
 1154 or optional.
 - 1155 • **Related Resource Types (optional)** – the specification of other Resource Types that may be
 1156 referenced as part of the Resource Type, applicable to collections.
 - 1157 • **Mime Types (optional)** – mime types supported by the resource including serializations (e.g.,
 1158 application/cbor, application/json, application/xml).

1159 Table 5 and Table 6 provide an example description of an illustrative foobar Resource Type and
 1160 its associated Properties.

1161

Table 5. Example foobar Resource Type

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

1162

Table 6. Example foobar properties

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource Type	rt	array			R	yes	Resource Type
Interface	if	array			R	yes	Interface
Foo value	value	string			R	yes	Foo value

1163

1164 An instance of the foobar Resource Type is as shown below

1165

```

{
  "rt": "oic.r.foobar",
  "if": "oic.if.a",
  "value": "foo value"
}

```

1166

1167

1168

1169 An example schema for the foobar Resource Type is shown below

1170

```

{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "rt": {"type": "string"},
    "if": {"type": "string"},
    "value": {"type": "string"}
  },
  "required": ["rt", "if", "value"]
}

```

1171

1172

1173

7.4.4

1175 Multi-valued Resource
1176 associations

Resource is
Resource

1177 Type IDs (e.g. "rt": ["oic.r.switch.binary", "oic.r.light.brightness"]). The order of the Resource Type
1178 IDs in the "rt" Property Value is meaningless. For example, "rt": ["oic.r.switch.binary",
1179 "oic.r.light.brightness"] and "rt": ["oic.r.light.brightness", "oic.r.switch.binary"] have the same
1180 meaning.

1181 Resource Types for multi-value "rt" Resources shall satisfy the following conditions.

- 1182 • **Property Name** – Property Names for each Resource Type shall be unique (within the scope
1183 of the multi-valued "rt" Resource) with the exception of Common Properties, otherwise there will
1184 be conflicting Property semantics. If two Resource Types have a Property with the same
1185 Property Name, a multi-valued "rt" Resource shall not be composed of these Resource Types.

1186 A multi-valued "rt" Resource satisfies all the requirements for each Resource Type and conforms to
1187 the RAML/JSON definitions for each component Resource Type. Thus the mandatory Properties
1188 of a multi-valued "rt" Resource shall be the union of all the mandatory Properties of each Resource
1189 Type. For example, mandatory Properties of a Resource with "rt": ["oic.r.switch.binary",

1190 "oic.r.light.brightness"] are "value" and "brightness", where the former is mandatory for
1191 "oic.r.switch.binary" and the latter for "oic.r.light.brightness".

1192 The multi-value "rt" Resource Interface set shall be the union of the sets of interfaces from the
1193 component Resource Types. The Resource Representation in response to a CRUDN action on an
1194 Interface shall be the union of the schemas that are defined for that Interface. The Default Interface
1195 for a multi-value "rt" Resource shall be the baseline Interface (oic.if.baseline) as that is the only
1196 guaranteed common Interface between the Resource Types.

1197 For clarity if each Resource Type supports the same set of Interfaces, then the resultant multi-
1198 value "rt" Resource has that same set of Interfaces with a Default Interface of baseline
1199 (oic.if.baseline).

1200 An "rt" query for a multi-value "rt" Resource with the Default Interface of "oic.if.a", "oic.if.s", "oic.if.r",
1201 "oic.if.rw" or "oic.if.baseline" is an extension of a generic "rt" query. When a Server receives a
1202 RETRIEVE request for multi-value "rt" Resource with an "rt" query, (i.e. GET
1203 /ResExample?rt=oic.r.foo), the Server should respond only when the query value is an item of the
1204 "rt" Property Value of the target Resource and should send back only the Properties associated
1205 with the query value. For example, upon receiving GET /ResExample?rt=oic.r.switch.binary
1206 targeting a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"], the Server responds
1207 with only the Properties of oic.r.switch.binary.

1208

1209 **7.5 Device Type**

1210 A Device Type is a class of Device. Each Device Type defined will include a list of minimum
1211 Resource Types that a device shall implement for that Device Type. A device may expose
1212 additional standard and vendor defined Resource Types beyond the minimum list. The Device
1213 Type is used in Resource discovery as specified in section 11.3.4.

1214 Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in
1215 a Link using the Resource Type Parameter.

1216 A Device Type may either be pre-defined (in vertical domain specifications) or in custom definitions
1217 by manufacturers, end users, or developers of Devices (vendor-defined Device Types). Device
1218 Types and their definition details may be communicated out of band (like in documentation).

1219 Every Device Type shall be identified with a Resource Type ID using the same syntax constraints
1220 as a Resource Type.

1221 **7.6 Interface**

1222 **7.6.1 Introduction**

1223 An Interface provides first a view into the Resource and then defines the requests and responses
1224 permissible on that view of the Resource. So this view provided by an Interface defines the context
1225 for requests and responses on a Resource. Therefore, the same request to a Resource when
1226 targeted to different Interfaces may result in different responses.

1227 An Interface may be defined by either this specification (a Core Interface), the OCF vertical domain
1228 specifications (a "vertical Interface) or manufacturers, end users or developers of Devices (a
1229 "vendor-defined Interface").

1230 The Interface Property lists all the Interfaces the Resource support. All resources shall have at
1231 least one Interface. The Default Interface shall be defined by an OCF specification and inherited
1232 from the Resource Type definition. The Default Interface associated with all Resource Types
1233 defined in this specification shall be the supported Interface listed first within the applicable

1234 enumeration in the definition of the Resource Type (see Annex D). All Default Interfaces specified
 1235 in an OCF specification shall be mandatory.

1236 In addition to any OCF specification defined interface, all Resources shall support the Baseline
 1237 Interface (oic.if.baseline) as defined in section 7.6.3.2.

1238 When an Interface is to be selected for a Request, it shall be specified as query parameter in the
 1239 URI of the Resource in the Request message. If no query parameter is specified, then the Default
 1240 Interface shall be used. If the selected Interface is not one of the permitted Interfaces on the
 1241 Resource then selecting that Interface is an error.

1242 An Interface may accept more than one media type. An Interface may respond with more than one
 1243 media type. The accepted media types may be different from the response media types. The media
 1244 types are specified with the appropriate header parameters in the transfer protocol. (NOTE: This
 1245 feature has to be used judiciously and is allowed to optimize representations on the wire) Each
 1246 Interface shall have at least one media type.

1247

1248 7.6.2 Interface Property

1249 **Table 7. Resource Interface Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface	if	json	Array of Dot separated strings		R	yes	Property to declare the Interfaces supported by a Resource.

1250 The Interfaces supported by a Resource shall be declared using the Interface Common Property
 1251 (Table 7) as "if=<array of Interfaces>". The Property Value of an Interface Property shall be a
 1252 lower case string with segments separated by a "." (dot). The string "oic", when used as the first
 1253 segment in the Interface Property Value, is reserved for OCF-defined Interfaces. The Interface
 1254 Property Value may also be a reference to an authority similar to IANA that may be used to find
 1255 the definition of an Interface. A Resource Type shall support one or more of the Interfaces defined
 1256 in section 7.6.3.

1257 7.6.3 Interface methods

1258 7.6.3.1 Overview

1259 The OCF -defined Interfaces are listed in the table below:

1260 **Table 8. OCF standard Interfaces**

Interface	Name	Applicable Methods	Description
baseline	oic.if.baseline	RETRIEVE, UPDATE	The baseline Interface defines a view into all Properties of a Resource including the Meta Properties. This Interface is used to operate on the full Representation of a Resource.
links list	oic.if.ll	RETRIEVE	The 'links list' Interface provides a view into Links in a Collection (Resource). Since Links represent relationships to other Resources, the links list 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 Interface to allow discovery of Resource "hosted" on a Device.

batch	oic.if.b	RETRIEVE, UPDATE	The batch 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	The read-only Interface exposes the Properties of a Resource that may be 'read'. This Interface does not provide methods to update Properties or a Resource and so can only be used to 'read' Property Values.
read-write	oic.if.rw	RETRIEVE, UPDATE	The read-write Interface exposes only those Properties that may be both 'read' and "written" and provides methods to read and write the Properties of a Resource.
actuator	oic.if.a	CREATE, RETRIEVE, UPDATE	The actuator Interface is used to read or write the Properties of an actuator Resource.
sensor	oic.if.s	RETRIEVE	The sensor Interface is used to read the Properties of a sensor Resource.

1261

1262

1263 7.6.3.2 Baseline Interface

1264 7.6.3.2.1 Overview

1265 The Representation that is visible using the "baseline" Interface includes all the Properties of the
1266 Resource including the Common Properties. The "baseline" Interface shall be defined for all
1267 Resource Types. All Resources shall support the "baseline" Interface.

1268 The "baseline" Interface is selected by adding if=oic.if.baseline to the list of query parameters in
1269 the URI of the target Resource. For example: GET /oic/res?if=oic.if.baseline.

1270 7.6.3.2.2 Use of RETRIEVE

1271 The "baseline" Interface is used when a Client wants to retrieve all Properties of a Resource. The
1272 Client includes the URI query parameter definition "?if=oic.if.baseline" in a RETRIEVE request.
1273 When this query parameter definition is included the Server shall respond with a Resource
1274 representation that includes all of the implemented Properties of the Resource. When the Server
1275 is unable to send back the whole Resource representation, it shall reply with an error message.
1276 The Server shall not return a partial Resource representation.

1277 An example response to a RETRIEVE request using the baseline Interface is shown below:

```
{
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a","oic.if.baseline"],
  "temperature": 20,
  "units": "C",
  "range": [0,100]
}
```

1278

1279 7.6.3.2.3 Use of UPDATE

1280 Using the baseline Interface, all Properties of a Resource with the exception of Common Properties
1281 may be modified using an UPDATE request with a list of Properties and their desired values if a
1282 Resource Type has an associated schema for UPDATE using baseline.

1283 **7.6.3.3 Link List Interface**

1284 **7.6.3.3.1 Overview**

1285 The links list Interface provides a view into the list of Links in a Collection (Resource). The
1286 Representation visible through this Interface has only the Links defined in the Property Value of
1287 the “links” Property – so this Interface is used to manipulate or interact with the list of Links in a
1288 Collection. The Links list may be RETRIEVED using this Interface.

1289 The Interface definition and semantics are given as follows:

- 1290 • The links list Interface name shall be “oic.if.ll”.
- 1291 • If specified in a request (usually in the request header), the serialization in the response shall
1292 be in the format expected in the request.
- 1293 • In response to a RETRIEVE request on the “links list” Interface, the URIs of the referenced
1294 Resources shall be returned as a URI reference.
- 1295 • If there are no links present in a Resource, then an empty list shall be returned.
- 1296 • The Representation determined by this Interface view only includes the Property Value of the
1297 “links” Property.

1298 **7.6.3.3.2 Example: “links list” Interface**

1299 **Example: Request to a Collection**

<p>Request to RETRIEVE the Links in room</p> <p>(the Links could be referencing lights, fans, electric sockets etc)</p>	<pre>GET ocf://<devID>/a/room/1?if=oic.if.ll</pre>
--	--

1300

1301 **7.6.3.4 Batch Interface**

1302 **7.6.3.4.1 Overview**

1303 The batch Interface is used to interact with a collection of Resources using a single/same Request.
1304 The batch Interface supports methods of Resources in the Links of the Collection, and can be used
1305 to RETRIEVE or UPDATE the Properties of the “linked” Resources with a single Resource
1306 representation.

1307 The batch Interface selects a view into the Links in a Collection – the Request is sent to all the
1308 Links in this view with potential modifications defined in the Parameters of the Link

1309 The batch Interface is defined as follows:

- 1310 • The batch Interface name shall be “oic.if.b”
- 1311 • A Collection Resource with a batch Interface has Links that have Resource references that
1312 may be URIs (fully qualified for remote Resources) or relative references (for local Resources).
- 1313 • The original request is modified to create new requests targeting each of the targets in the
1314 Resource Links by substituting the URI in the original request with the URI of the target
1315 Resource in the Link. The payload in the original request is replicated in the payload of the
1316 new Requests.
- 1317 • The Requests shall be forwarded assuming use of the Default Interface of the referenced
1318 Resources.

- 1319 • Requests shall only be forwarded to link targets that are identified as items in the collection by
1320 relation types "item" or "hosts" ("hosts" is the default relation type). Requests shall not be
1321 forwarded to targets of links that do not contain the "item" or "hosts" relation type values. The
1322 default relation type "hosts" shall be allowed for relative and absolute links.
- 1323 • Resources of the collection itself may be included in the batch by using the link relation "self"
1324 along with "item" and insuring that the default interface of the collection does not expose the
1325 links property, i.e. not oic.if.baseline or oic.if.ll.
- 1326 • All the Responses from the linked item Resources shall be aggregated into single Response
1327 to the Client. The Server may timeout the Response to a time window (if a time window has
1328 been negotiated with the Client then the Server shall not timeout within that window; in the
1329 absence of negotiated window, the Server may choose any appropriate window based on
1330 conditions). If the target Resources cannot process the new request, an empty response or
1331 error response shall be returned. These empty/error Responses shall be included in
1332 aggregated Response to the original Client Request.
- 1333 • The batch representation is an array of objects representing the responses from the linked
1334 resources. Each object in the batch response shall include at least two items: (1) the URI of
1335 the linked resource (fully qualified for remote resources, or a relative reference for local
1336 resources) as "href": <URI> and (2) the individual response object as "rep" as the key i.e. "rep":
1337 { < representation of individual response > }.
- 1338 • Resources referenced by links in the collection may be observed using the batch interface of
1339 the collection. The observe mechanism shall work as defined in 11.4.2. Specifically, the
1340 representations and status codes shall be the same as for RETRIEVE operations using the
1341 Batch interface.
- 1342 • Properties of the collection resource itself may be observed by using the appropriate interface.
1343 For example, a collection may be observed on its linked list or baseline interface to receive
1344 notifications of changes to its links.
- 1345 • The Client may choose to restrict the list of Links to which the Request is forwarded by including
1346 query parameters in the URI of the Collection to which this original 'batch' Interface Request
1347 is made. The Server should process query parameters in a request that includes oic.if.b, as
1348 selectors for links in the Collection that are to be processed in the batch.
- 1349 • Batch UPDATE operations are performed by creating a payload according to the same schema
1350 of the Batch RETRIEVE payload. A set of link-specific UPDATE requests is created according
1351 to the "href" tags in the included items, and the payload contained in the value of the "rep"
1352 property is applied to the corresponding "href" referenced item.
- 1353 • If requested property for UPDATE does not exist in linked resource, it shall silently ignore the
1354 request.
- 1355 • If the "href" value is the empty URI, denoted by a zero length string or "" in JSON, the payload
1356 in the value of the "rep" property is applied to all batch items in the Collection.
- 1357 • Items with the empty "href" and link-specific "href" shall not be mixed in the same UPDATE
1358 payload.
- 1359 • The Representation in the Link-specific Request may not match the Representation exposed
1360 by the Default Interface on the target Resource. In such cases, the 'subset' semantics apply
1361 where Properties in the Request which match Properties in the Resource view exposed shall
1362 be modified in the target Resource if the Property is writeable.
- 1363 • The response to POST shall contain the updated values using the same payload schema as
1364 RETRIEVE operations, along with the appropriate status code. The response payload shall
1365 reflect the known state of the updated resource properties after the batch update was
1366 completed.

1367 **7.6.3.4.2 Use of Query Parameters with Batch**

1368 Additional query parameters may be used with the batch interface in order to select particular items
1369 in the batch for retrieval or update. When additional parameters are included which are not
1370 interpreted in other ways, these parameters are used to select items in the batch by matching link
1371 attribute values.

1372 A particular item in a batch is selected by additional query parameters in a request if, and only if,
1373 all of the link selection query parameters contained values which match corresponding values in
1374 the link attributes of that item.

1375 When link selection query parameters are used with RETRIEVE operations, only the items with
1376 matching link attributes are returned.

1377 When link selection query parameters are used with UPDATE operations, only the items having
1378 matching link attributes are updated.

1379 See 7.6.3.4.3 for examples of RETRIEVE and UPDATE operations that use link selection query
1380 parameters.

1381 **7.6.3.4.3 Examples: Batch Interface**

1382 Example 1

Resources	<pre>/a/room/1 { "rt": ["oic.wk.col", "x.org.example.rt.room"], "if": ["oic.if.baseline", "oic.if.b", "oic.if.ll", "oic.if.r"], "x.org.example.color": "blue", "x.org.example.dimension": "15bx15wx10h", "links": [{ "href": "/a/room/1", "rel": ["self", "item"], "rt": ["x.org.example.rt.room"], "if": ["oic.if.r", "oic.if.baseline", "oic.if.b", "oic.if.ll"] }, { "href": "/the/light/1", "rel": ["item"], "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"] }, { "href": "/the/light/2", "rel": ["item"], "rt": ["oic.r.switch.binary"], if=["oic.if.a", "oic.if.baseline"] }, { "href": "/my/fan/1", "rel": ["item"], "rt": ["oic.r.switch.binary"], if=["oic.if.a", "oic.if.baseline"] }, { "href": "/his/fan/2", "rel": ["item"], "rt": ["oic.r.switch.binary"], if=["oic.if.a", "oic.if.baseline"] }] }</pre>
-----------	---

```
/the/light/1
{
  "rt": ["oic.r.switch.binary "],
  "ins": "light-1",
  "if": ["oic.if.a", "oic.if.baseline"],
  "value": false
}

/the/light/2
{
  "rt": ["oic.r.switch.binary "],
  "ins": "light-2",
  "if": ["oic.if.a", "oic.if.baseline"],
  "value": true
}

/the/fan/1
{
  "rt": ["oic.r.switch.binary "],
  "ins": "fan-1",
  "if": ["oic.if.a", "oic.if.baseline"],
  "value": true
}

/the/fan/2
{
  "rt": ["oic.r.switch.binary "],
  "ins": "fan-2",
  "if": ["oic.if.a", "oic.if.baseline"],
```

	<pre>"value": false }</pre>
Use of batch	<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 Interface as specified for this resource) GET /the/light/1 (NOTE: Uses the default Interface as specified for this resource) GET /the/light/2 (NOTE: Uses the default Interface as specified for this resource) GET /the/fan/1 (NOTE: Uses the default Interface as specified for this resource) GET /the/fan/2 (NOTE: Uses the default Interface as specified for this resource) Response: [{ "href": "/a/room/1", "rep": {"x.org.example.color": "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} }]</pre>

	<pre> }, { "href": "/his/fan/2", "rep": {"value": false} }] </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 batch update payload item is the empty URI, the request is forwarded to all items in the batch 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 } </pre> <p>The response will be same as response for GET /a/room/1?if=oic.if.b.</p> <p>Since /a/room/1 does not have a "value" property exposed by its default interface, the update request will be silently ignored.</p>
<p>Use of batch</p> <p>(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.color": "red" } }] </pre>

```
]
```

This turns /the/light/1 off, turns /the/light/2 on, and sets the color of the room to "red".

The response will be same as response for GET /a/room/1?if=oic.if.b.

Example use of additional query parameters to select items by matching link attributes.

Turn on light 1 based on the "ins" link attribute value of "light-1"

```
UPDATE /a/room/1?if=oic.if.b&ins=light-1
```

```
[
  {
    "href": "",
    "rep": {
      "value": false
    }
  }
]
```

Similar to the earlier example, "href": "" applies the payload to all selected links. Since the additional query parameter ins=light-1 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.

Retrieving the item using the same query parameter:

```
RETRIEVE /a/room/1?if=oic.if.b&ins=light-1
```

Response payload:

```
[
  {
    "href": "",
    "rep": {
      "value": false
    }
  }
]
```

1383

1384 Example that further shows the "links list" and "batch" interface

Example

```
/myexample
{
  "rt": ["oic.r.foo"],
  "if": [ "oic.if.baseline", "oic.if.ll" ],
  "links": [
    { "href": "/acme/switch", "di": "<deviceID1>", "rt":
["oic.r.switch.binary"], "if": ["oic.if.a"] },
    { "href": "ocf://<deviceID1>/acme/fan", "rt":
["oic.r.fan"], "if": ["oic.if.a"] }
  ]
}
```

Use of Baseline	<pre>GET /myexample?if=oic.if.baseline will return { "rt": ["oic.r.foo"], "if": ["oic.if.baseline", "oic.if.ll"], "links": [{ "href": "/acme/switch", "di": "<deviceID1>", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a"]}, { "href": "ocf://<deviceID1>/acme/fan", "rt": "oic.r.fan", "if": "oic.if.a"}] }</pre>
Use of Links List	<pre>GET /myexample?if=oic.if.ll. will return [{ "href": "/acme/switch", "di": "<deviceID1>", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a"]}, { "href": "ocf://<deviceID1>/acme/fan", "rt": ["oic.r.fan"], "if": ["oic.if.a"]}]</pre>

1385

1386 **7.6.3.5 Actuator Interface**

1387 The actuator Interface is the Interface for viewing Resources that may be actuated i.e. changes
1388 some value within or the state of the entity abstracted by the Resource:

- 1389
- The actuator Interface name shall be “oic.if.a”
 - The actuator Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable JSON; the actuator interface may also expose in the Resource Representation optional Properties as defined by the applicable JSON schema that are implemented by the target Device.
- 1390
1391
1392
1393

For the following Resource

NOTE: “prm” is the Property name for ‘parameters’ Property

```
/a/act/heater
{
  "rt": ["acme.gas"],
  "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
  "prm": {"sensitivity": 5, "units": "C", "range": "0 .. 10"},
  "settemp": 10,
  "currenttemp" : 7
}
```

1394

Figure 8: Example - "Heater" Resource (for illustration only)

1395

NOTE: The example here is with respect to Figure 8

1. Retrieving values of an actuator

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

```

Response:
  {
    "prm": {"sensitivity": 5, "units": "C", "range": "0 .. 10"},
    "settemp": 10,
    "currenttemp" : 7
  }

2. Correct use of actuator:

Request: POST /a/act/heater?if="oic.if.a"
  {
    "settemp": 20
  }
Response:
  {
    Ok
  }

3. Incorrect use of actuator

Request: POST /a/act/heater?if="oic.if.a"
  {
    "if": "oic.if.s" ← this is visible through baseline
  }
Interface
Response:
  {
    Error
  }

```

Figure 9: Example - Actuator Interface

1396

- 1397 • A RETRIEVE request using this Interface shall return the Representation for this Resource
- 1398 subject to any query and filter parameters that may also exist
- 1399 • An UPDATE request using this Interface shall provide a payload or body that contains the
- 1400 Properties that will be updated on the target Resource.

1401 **7.6.3.6 Sensor Interface**

1402 The sensor Interface is the Interface for retrieving measured, sensed or capability specific
 1403 information from a Resource that senses:

- 1404 • The sensor Interface name shall be "oic.if.s"
- 1405 • The sensor Interface shall expose in the Resource Representation all mandatory Properties as
- 1406 defined by the applicable JSON; the sensor interface may also expose in the Resource
- 1407 Representation optional Properties as defined by the applicable JSON schema that are
- 1408 implemented by the target Device.
- 1409 • A RETRIEVE request using this Interface shall return this Representation for the Resource
- 1410 subject to any query and filter parameters that may also exist
- 1411 •

NOTE: The example here is with respect to Figure 8

1. Retrieving values of sensor

```

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

Response:
  {
    "currenttemp": 7
  }

2. Incorrect use of sensor

Request: PUT /a/act/heater?if="oic.if.s" ← PUT is not allowed
  {
    "settemp": 20 ← this is possible through actuator Interface
  }
Response:
  {
    Error
  }

3. Incorrect use of sensor

Request: POST /a/act/heater?if="oic.if.s" ← POST is not allowed
  {
    "currenttemp": 15 ← this is possible through actuator
Interface
  }
Response:
  {
    Error
  }

```

1412

1413 7.6.3.7 Read-only Interface

1414 The read-only Interface exposes only the Properties that may be “read”. This includes Properties
 1415 that may be “read-only”, “read-write” but not Properties that are “write-only” or “set-only”. The
 1416 applicable methods that can be applied to a Resource is RETRIEVE only. An attempt by a Client
 1417 to apply a method other than RETRIEVE to a Resource shall be rejected with an error response
 1418 code.

1419 7.6.3.8 Read-write Interface

1420 The read-write Interface exposes only the Properties that may be “read” and “written”. The “read-
 1421 only” Properties shall not be included in Representation for the “read-write” Interface. This is a
 1422 generic Interface to support “reading” and “setting” Properties in a Resource. The applicable
 1423 methods that can be applied to a Resource are RETRIEVE and UPDATE only. An attempt by a
 1424 Client to apply a method other than RETRIEVE or UPDATE to a Resource shall be rejected with
 1425 an error response code.

1426 7.7 Resource representation

1427 Resource representation captures the state of a Resource at a particular time. The resource
 1428 representation is exchanged in the request and response interactions with a Resource. A Resource
 1429 representation may be used to retrieve or update the state of a resource.

1430 The resource representation shall not be manipulated by the data connectivity protocols and
 1431 technologies (e.g., CoAP, UDP/IP or BLE).

1432 7.8 Structure

1433 7.8.1 Introduction

1434 In many scenarios and contexts, the Resources may have either an implicit or explicit structure
1435 between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The
1436 Framework provides the means to model and map these structures and the relationships among
1437 Resources. The primary building block for resource structures in Framework is the collection. A
1438 collection represents a container, which is extensible to model complex structures.

1439 7.8.2 Resource Relationships

1440 Resource relationships are expressed as Links. A Link embraces and extends typed web links
1441 concept as a means of expressing relationships between Resources. A Link consists of a set of
1442 Parameters that define:

- 1443 • a context URI,
- 1444 • a target URI,
- 1445 • a relation from the context URI to the target URI
- 1446 • elements that provide metadata about the target URI, the relationship or the context of the Link.

1447 The target URI is mandatory and the other items in a Link are optional. Additional items in the Link
1448 may be made mandatory based on the use of the links in different contexts (e.g. in collections, in
1449 discovery, in bridging etc.). Schema for the Link payload is provided in Annex D.

1450 An example of a Link is shown in

```
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "/room2"oic.if.baseline"], "p":  
{"bm": 3}, "rel": "item" }
```

1451 **Figure 10: Example of a Link**

1452 Two Links are distinct from each other when at least one parameter is different. For example the
1453 two Links shown in Figure 11 are distinct and can appear in the same list of Links.

```
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm":  
2}, "rel": "item" }  
  
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm":  
2}}
```

1454 **Figure 11: Example of distinct Links**

1455 The specification may mandate Parameters and Parameter values as required for certain
1456 capabilities. For all Links returned in a response to a RETRIEVE on /oic/res, if a Link does not
1457 explicitly include the "rel" Parameter, a value of "rel"="hosts" shall be assumed. The relation value
1458 of "hosts" is defined by IETF RFC 6690, the value of "item" by IETF RFC 6573, and the value of
1459 "self" by IETF RFC 4287 and all are registered in the IANA Registry for Link Relations at
1460 [\[http://www.iana.org/assignments/link-relations/link-relations.xhtml\]](http://www.iana.org/assignments/link-relations/link-relations.xhtml)

1461 As shown in D.2.8 the relation between the context URI and target URI in a Link is specified using
1462 the "rel" JSON element and the value of this element specifies the particular relation.

1463 The context URI of the Link shall implicitly be the URI of the Resource (or specifically a Collection)
1464 that contains the Link unless the Link specifies the anchor parameter. The anchor parameter is
1465 used to change the context URI of a Link – the relationship with the target URI is based off the

1466 anchor URI when the anchor is specified. Anchor parameter uses transfer protocol URI for OIC 1.1
 1467 Link (e.g. "anchor": "coaps://[fe80::b1d6]:44444") and OCF URI defined in Sec 6 for OCF 1.0 Links
 1468 (e.g. "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989").

1469 An example of using anchors in the context of Collections – a floor has rooms and rooms have
 1470 lights – the lights may be defined in floor as Links but the Links will have the anchor set to the URI
 1471 of the rooms that contain the lights (the relation is contains). This allows all lights in a floor to be
 1472 turned on or off together while still having the lights defined with respect to the rooms that contain
 1473 them (lights may also be turned on by using the room URI too).

```

/a/floor {
  "links": [
    {
      "href": "/x/light1",
      "anchor": "/a/room1",    ** Note: /a/room1 has the "item" relationship with /x/light1;
not /a/floor **
      "rel": "item"
    }
  ]
}

/a/room1 {
  "links": [
    {
      ** Note: /a/room1 "contains" the /x/light since /a/room1 is the implicit context URI **
      "href": "/x/light1",
      "rel": "item"
    }
  ]
}
  
```

1474 **Figure 12: Example of use of anchor in Link**

1475 **7.8.2.1 Parameters**

1476 **7.8.2.1.1 “ins” or Link Instance Parameter**

1477 The “ins” parameter identifies a particular Link instance in a list of Links. The "ins" parameter may
 1478 be used to modify or delete a specific Link in a list of Links. The value of the “ins” parameter is set
 1479 at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has
 1480 been set, the “ins” parameter shall not be modified for as long as the Link is a member of that list.

1481 **7.8.2.1.2 “p” or Policy Parameter**

1482 The Policy Parameter defines various rules for correctly accessing a Resource referenced by a
 1483 target URI. The Policy rules are configured by a set of key-value pairs as defined below.

1484 The policy Parameter "p" is defined by:

- 1485 • “bm” key: The “bm” key corresponds to an integer value that is interpreted as an 8-bit bitmask.
 1486 Each bit in the bitmask corresponds to a specific Policy rule. The following rules are specified
 1487 for “bm”:
 1488

Bit Position	Policy rule	Comment
--------------	-------------	---------

Bit 0 (the LSB)	discoverable	<p>The discoverable rule defines whether the Link is to be included in the Resource discovery message via /oic/res.</p> <ul style="list-style-type: none"> • 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	<p>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 at [http://www.iana.org/assignments/link-relations/link-relations.xhtml].</p> <ul style="list-style-type: none"> • 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.

1489

1490 Note that if all the bits in “bm” are defined to value 0, then the “bm” key may be omitted entirely
1491 from “p” as an efficiency measure. However, if any bit is set to value 1, then “bm” shall be
1492 included in “p” and all the bits shall be defined appropriately.

1493 • "sec" and "port" in the remaining bullets shall be used only in an OIC 1.1 payload. In OCF 1.0
1494 payload, "sec" and "port" shall not be used and instead the "eps" Parameter shall provide the
1495 information for an encrypted connection.

1496 • "sec" key: The “sec” key corresponds to a Boolean value that indicates whether the Resource
1497 referenced by the target URI is accessed via an encrypted connection. If “sec” is true, the
1498 resource is accessed via an encrypted connection, using the “port” specified (see below). If
1499 “sec” is false, the resource is accessed via an unencrypted connection, or via an encrypted
1500 connection (if such a connection is made using the “port” settings for another Resource, for
1501 which “sec” is true).

1502 • "port" key: The “port” key corresponds to an integer value that is used to indicate the port
1503 number where the Resource referenced by the target URI may be accessed via an encrypted
1504 connection.

1505 • If the Resource is only available via an encrypted connection (i.e. DTLS over IP), then
1506 ○ "p" shall include the "sec" key and its value shall be true.

- 1507 o "p" shall include the "port" key and its value shall be the port number where the
1508 encrypted connection may be established.
- 1509 • If the Resource is not available via an encrypted connection, then
- 1510 o "p" shall include the "sec" key and its value shall be false or "p" shall omit the "sec"
1511 key; the default value of "sec" is false.
- 1512 o "p" shall omit the "port" key.
- 1513 o A Resource that is available via either an encrypted or unencrypted connection
1514 follows the population scheme defined in this clause.
- 1515 • Access to the Resource on the port specified by the "port" key shall be made by an encrypted
1516 connection (e.g. coaps://). (Note that unencrypted connection to the Resource may be possible
1517 on a separate port discovered thru multicast discovery).
- 1518 • Note that access to the Resource is controlled by the ACL for the Resource. A successful
1519 encrypted connection does not ensure that the requested action will succeed. See
1520 OCF Security – Access Control section for more information.

1521 Example 1: below shows the Policy Parameter for a Resource that is discoverable but not
1522 observable, and for which authenticated accesses shall be done via CoAPS port 33275::

```
1523 "p": { "bm": 2 }
```

1524

1525 Example 2: below shows a self-link, i.e. the /oic/res Link in itself that is discoverable and
1526 observable.

```
1527 {  
1528   "href": "/oic/res",  
1529   "rel": "self",  
1530   "rt": ["oic.wk.res"],  
1531   "if": ["oic.if.ll", "oic.if.baseline"],  
1532   "p": { "bm": 3 }  
1533 }
```

1534 **7.8.2.1.3 “type” or Media Type Parameter**

1535 The “type” Parameter may be used to specify the various media types that are supported by a
1536 specific target Resource. The default type of "application/cbor" shall be used when the “type”
1537 element is omitted. Once a Client discovers this information for each Resource, it may use one of
1538 the available representations in the appropriate header field of the Request or Response.

1539 **7.8.2.1.4 “bp” or the Batch Interface Parameter**

1540 The “batch” Parameter "bp" is used to specify the modifications to the target URI as the "batch"
1541 Request is forwarded through this Link. The "q" element in the value defines the query string that
1542 shall be appended to the "href" to make the target URI. The "q" query string may contain Property
1543 strings that are valid in that context. For example: Given a Collection as follows

```
/room2  
{  
  "if": "oic.if.b",  
  "colour": "blue",
```

```
"links": [
  { "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p":
    {"bm": 2}, "rel": "contains", "bp": { "q": "if=oic.if.baseline" }
  }
]
```

1544 The following is the sequence for batch request to /room2

1. GET /room2?if=oic.if.b
2. This request is transformed to: GET /switch?if=oic.if.baseline when the batch request is propagated through the Link to the target /switch

1545 See the Interfaces section 7.5 for more details on the "batch" Interface.

1546 7.8.2.1.5 "di" or Device ID parameter

1547 The "di" Parameter specifies the device ID of the Device that hosts the target Resource defined in
1548 the in the "href" Parameter.

1549 The device ID may be used to qualify a relative reference used in the "href" or to lookup endpoint
1550 information for the relative reference.

1551 7.8.2.1.6 "eps" Parmeter

1552 The "eps" Parameter indicates the Endpoint information of the target Resource.

1553 "eps" shall have as its value an array of items and each item represents Endpoint information with
1554 "ep" and "pri" as specified in 10.2. "ep" is mandatory but "pri" is optional.

1555 Figure 13 is illustrated for "eps" with multiple Endpoints.

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

1556 **Figure 13: Example of "eps Parameter**

1557 When "eps" is present in a link, the Endpoint information in "eps" can be used to access the target
1558 Resource referred by the "href" Parameter.

1559 When present, max-age information (e.g. Max-Age option for CoAP defined in IETF RFC 7252)
1560 determines the maximum time "eps" values may be cached before they are considered stale.

1561 7.8.2.2 Formatting

1562 When formatting in JSON, the list of Links shall be an array.

1563 7.8.2.3 List of Links in a Collection

1564 A list of Links in a Resource shall be included in that Resource as the value of the "links" Property
1565 of that Resource. A Resource that contains Links is a Collection.

```

/Room1
{
  "rt": "my.room",
  "if": [ "oic.if.ll", "oic.if.baseline" ],
  "color": "blue",
  "links":
  [
    {
      "href": "/oic/d",
      "rt": [ "oic.d.light", "oic.wk.d" ],
      "if": [ "oic.if.r", "oic.if.baseline" ],
      "p": { "bm": 1 }
    },
    {
      "href": "/oic/p",
      "rt": [ "oic.wk.p" ],
      "if": [ "oic.if.r", "oic.if.baseline" ],
      "p": { "bm": 1 }
    },
    {
      "href": "/switch",
      "rt": [ "oic.r.switch.binary" ],
      "if": [ "oic.if.a", "oic.if.baseline" ],
      "p": { "bm": 3 },
      "mt": [ "application/cbor", "application/exi+xml" ]
    },
    {
      "href": "/brightness",
      "rt": [ "oic.r.light.brightness" ],
      "if": [ "oic.if.a", "oic.if.baseline" ],
      "p": { "bm": 3 }
    }
  ]
}

```

Figure 14: List of Links in a Resource

1567

1568 7.8.3 Collections

1569 7.8.3.1 Overview

1570 A Resource that contains one or more references (specified as Links) to other resources is an
 1571 Collection. These reference may be related to each other or just be a list; the Collection provides
 1572 a means to refer to this set of references with a single handle (i.e. the URI). A simple resource is
 1573 kept distinct from a collection. Any Resource may be turned into an Collection by binding resource
 1574 references as Links. Collections may be used for creating, defining or specifying hierarchies,
 1575 indexes, groups, and so on.

1576 A Collection shall have at least one Resource Type and at least one Interface bound at all times
 1577 during its lifetime. During creation time of a collection the Resource Rype and interfaces are
 1578 specified. The initial defined Resource Types and interfaces may be updated during its life time.
 1579 These initial values may be overridden using mechanism used for overriding in the case of a
 1580 Resource. Additional Resource Types and Interfaces may be bound to the Collection at creation
 1581 or later during the lifecycle of the Collection.

1582 A Collection shall define the “links” Common Property. The value of the “links” Property is an array
1583 with zero or more Links. The target URIs in the Links may reference another Collection or another
1584 Resource. The referenced Collection or Resource may reside on the same Device as the Collection
1585 that includes that Link (called a local reference) or may reside on another Device (called a remote
1586 reference). The context URI of the Links in the “links” array shall (implicitly) be the Collection that
1587 contains that “links” property. The (implicit) context URI may be overridden with explicit
1588 specification of the “anchor” parameter in the Link where the value of “anchor” is the new base of
1589 the Link.

1590 A Resource may be referenced in more than one Collection, therefore, a unique parent-child
1591 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the
1592 Resource referenced in the Collection, i.e., the application may use Collections to represent a
1593 relationship but none is automatically implied or defined. The lifecycles of the Collection and the
1594 referenced Resource are also independent of one another.

1595 If the “drel” property is defined for the Collection then all Links that don’t explicitly specify a
1596 relationship shall inherit this default relationship in the context of that Collection. The default
1597 relationship defines the implicit relationship between the Collection and the target URI in the Link.

1598 A Property "links" represents the list of Links in a Collection. "links" Property has, as its value, an
1599 array of items and each item is an OCF Link as shown in Figure 15.



1601 **Figure 15: Example showing Collection and Links**

1603 A Collection may be:

- 1604 • A pre-defined Collection where the Collection has been defined a priori and the Collection is
1605 static over its lifetime. Such Collections may be used to model, for example, an appliance that
1606 is composed of other devices or fixed set of resource representing fixed functions.
- 1607 • A Device local Collection where the Collection is used only on the Device that hosts the
1608 Collection. Such collections may be used as a short-hand on a client for referring to many
1609 Servers as one.
- 1610 • A centralized Collection where the Collection is hosted on an Device but other Devices may
1611 access or update the Collection
- 1612 • A hosted Collection where the collection is centralized but is managed by an authorized agent
1613 or party.

1614 7.8.3.2 Collection Properties

1615 An Collection shall define the “links” Property. In addition, other Properties may be defined for the
1616 Collection by the Resource Type. The mandatory and recommended Common Properties for
1617 Collection are shown in Table 9. This list of Common Properties are in addition to those defined
1618 for Resources in section 7.3.2. When a property is repeated in Table 9 , the conditions in this
1619 definition shall override those in the general list for Resources.

1620 **Table 9. Common Properties for Collections (in addition to Common Properties defined in**
1621 **section 7.3.2)**

Property	Description	Property name	Value Type	Mandatory
Links	The set of links in the collection	“links”	json Array of Links	Yes
Name	Human friendly name for the collection	“n”	string	No
Identity	The id of the collection	“id”	UUID	No
Resource Types	The list of allowed Resource Types for links in the collection. Requests for addition of links using link list or link batch interfaces will be validated against this list. If this property is not defined or is null string then any Resource Type is permitted	“rts”	json Array of Resource Type names	No
Default relationship	Specifies the default relationship to use for Links in the collection where the “rel” parameter has not been explicitly defined. It is permissible to have no “drel” property defined for the collection and the Links to also not have “rel” defined either. In such case, the use of the collection is, for example, as a random bag of links	“drel”	string	No

1622

1623 The Properties of a Collection may not be modified.

1624 **7.8.3.3 Default Resource Type**

1625 A default Resource Type, oic.wk.col, shall be available for Collections. This Resource Type shall
1626 be used only when another type has not been defined on the Collection or when no Resource Type
1627 has been specified at the creation of the Collection.

1628 The default Resource Type provides support for the Common Properties including the “links”
1629 Property. For the default Resource Type, the value of “links” shall be a simple array of Links.

1630 The default Resource Type shall support the ‘baseline’ and ‘links list’ Interfaces. The default
1631 Interface shall be the ‘links list’ Interface.

1632

1633 **7.9 Third (3rd) party specified extensions**

1634 This section describes how a 3rd party may add Device Types, Resource Types, 3rd party defined
1635 Properties to an existing or 3rd party defined Resource Type, 3rd party defined enumeration values
1636 to an existing enumeration and 3rd party defined parameters to an existing defined Property.

1637 A 3rd party may specify additional (non-OCF) Resources within an OCF Device. A 3rd party may
1638 also specify additional Properties within an existing OCF defined Resource Type. Further a 3rd
1639 party may extend an OCF defined enumeration with 3rd party defined values.

1640 A 3rd party defined Device Type may expose both 3rd party and OCF defined Resource Types. A
1641 3rd party defined Device Type must expose the mandatory Resources for all OCF Devices defined
1642 within this specification.

1643 A 3rd party defined Resource Type shall include any mandatory Properties defined in this
1644 specification and also any vertical specified mandatory Properties. All Properties defined within a
1645 3rd party defined Resource Type that are part of the OCF namespace that are not Common
1646 Properties as defined in this specification shall follow the 3rd party defined Property rules in Table
1647 10.

1648 The following table defines the syntax rules for 3rd party defined Resource Type elements. Within
1649 the table the term “Domain_Name” refers to a domain name that is owned by the 3rd party that is
1650 defining the new element.

1651

Table 10. 3rd party defined Resource elements

	Resource Element	Vendor Definition Rules
New 3 rd party defined Device Type	“rt” Property Value of /oic/d	x.<Domain_Name>.<resource identification>
New 3 rd party defined Resource Type	“rt” Property Value	x.<Domain_Name>.<resource identification>
New 3 rd party defined Property within the OCF namespace	Resource Property Name	x.<Domain_Name>.<property>
Additional 3 rd party defined values in an OCF specified enumeration	Enumeration Property Value	x.<Domain_Name>.<enum value>
Additional 3 rd party defined parameter in an OCF specified Property	Parameter key word	x.<Domain_Name>.<parameter keyword>

1652

1653 With respect to the use of the Domain_Name in this scheme the labels are reversed from how they
1654 appear in DNS or other resolution mechanisms. The 3rd party defined Device Type and Resource
1655 Type otherwise follow the rules defined in Section 7.4.2 Resource Type Property. 3rd party defined

1656 Resource Types should be registered in the IANA Constrained RESTful Environments (CoRE)
1657 Parameters registry.

1658 For example:

1659 x.com.samsung.galaxyphone.accelerator

1660 x.com.cisco.ciscorouterport

1661 x.com.hp.printerhead

1662 x.org.allseen.newinterface.newproperty

1663

1664 8 CRUDN

1665 8.1 Overview

1666 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for
1667 manipulating Resources. These operations are performed by a Client on the resources contained
1668 in n Server.

1669 On reception of a valid CRUDN operation n Server hosting the Resource that is the target of the
1670 request shall generate a response depending on the Interface included in the request; or based
1671 on the Default Interface for the Resource Type if no Interface is included.

1672 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in
1673 Table 11. A Device shall use CBOR as the default payload (content) encoding scheme for resource
1674 representations included in CRUDN operations and operation responses; a Device may negotiate
1675 a different payload encoding scheme (e.g, see in section 12.2.4 for CoAP messaging). The
1676 following subsections specify the CRUDN operations and use of the parameters. The type
1677 definitions for these terms will be mapped in the messaging section for each protocol.

1678

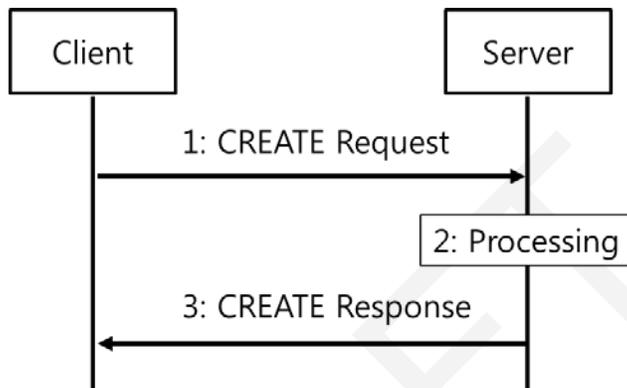
Table 11. Parameters of CRUDN messages

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 section 5.9 and 12.1.2 in IETF RFC 7252.
	<i>obs</i>	Observe	Indicator for an observe response.

1679 **8.2 CREATE**

1680 The CREATE operation is used to request the creation of new Resources on the Server. The
 1681 CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 16
 1682 and described below.



1683
 1684 **Figure 16. CREATE operation**

1685 **8.2.1 CREATE request**

1686 The CREATE request message is transmitted by the Client to the Server to create a new Resource
 1687 by the Server. The CREATE request message will carry the following parameters:

- 1688 • *fr*: Unique identifier of the Client
- 1689 • *to*: URI of the target resource responsible for creation of the new resource.
- 1690 • *ri*: Identifier of the CREATE request
- 1691 • *cn*: Information of the resource to be created by the Server
 - 1692 i) *cn* will include the URI and Resource Type property of the resource to be created.
 - 1693 ii) *cn* may include additional properties of the resource to be created.
- 1694 • *op*: CREATE

1695 **8.2.2 Processing by the Server**

1696 Following the receipt of a CREATE request, the Server may validate if the Client has the
 1697 appropriate rights for creating the requested resource. If the validation is successful, the Server
 1698 creates the requested resource. The Server caches the value of *ri* parameter in the CREATE
 1699 request for inclusion in the CREATE response message.

1700 **8.2.3 CREATE response**

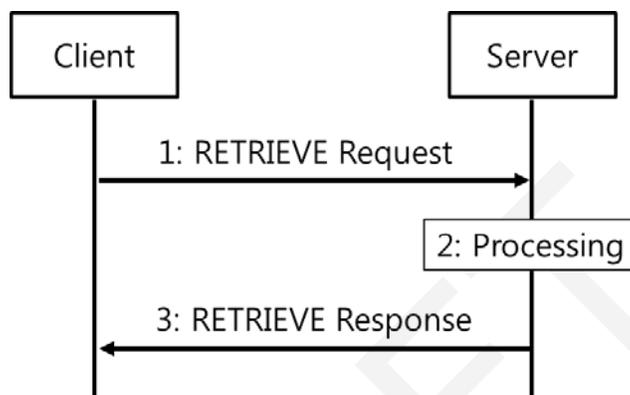
1701 The Server shall transmit a CREATE response message in response to a CREATE request
 1702 message from a Client. The CREATE response message will include the following parameters.

- 1703 • *fr*: Unique identifier of the Server
- 1704 • *to*: Unique identifier of the Client
- 1705 • *ri*: Identifier included in the CREATE request

- 1706 • *cn*: Information of the resource as created by the Server.
- 1707 i) *cn* will include the URI of the created resource.
- 1708 ii) *cn* will include the resource representation of the created resource.
- 1709 • *rs*: The result of the CREATE operation

1710 8.3 RETRIEVE

1711 The RETRIEVE operation is used to request the current state or representation of a Resource.
 1712 The RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in
 1713 Figure 17 and described below.



1714

1715 **Figure 17. RETRIEVE operation**

1716 8.3.1 RETRIEVE request

1717 RETRIEVE request message is transmitted by the Client to the Server to request the
 1718 representation of a Resource from a Server. The RETRIEVE request message will carry the
 1719 following parameters.

- 1720 • *fr*: Unique identifier of the Client
- 1721 • *to*: URI of the resource the Client is targeting
- 1722 • *ri*: Identifier of the RETRIEVE request
- 1723 • *op*: RETRIEVE

1724 8.3.2 Processing by the Server

1725 Following the receipt of a RETRIEVE request, the Server may validate if the Client has the
 1726 appropriate rights for retrieving the requested data and the properties are readable. The Server
 1727 caches the value of *ri* parameter in the RETRIEVE request for use in the response.

1728 8.3.3 RETRIEVE response

1729 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
 1730 message from a Client. The RETRIEVE response message will include the following parameters.

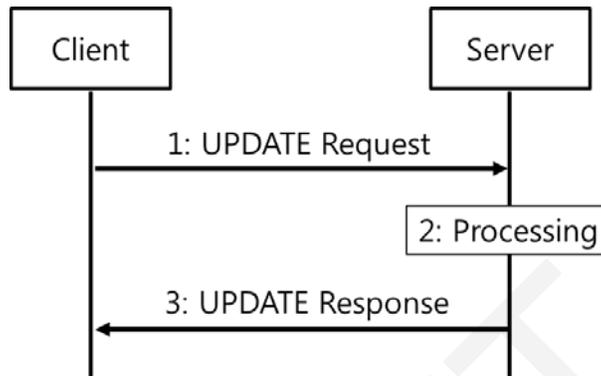
- 1731 • *fr*: Unique identifier of the Server
- 1732 • *to*: Unique identifier of the Client
- 1733 • *ri*: Identifier included in the RETRIEVE request
- 1734 • *cn*: Information of the resource as requested by the Client
- 1735 i) *cn* should include the URI of the resource targeted in the RETRIEVE request

1736

1737 • *rs*: The result of the RETRIEVE operation

1738 8.4 UPDATE

1739 The UPDATE operation is either a Partial UPDATE or a complete replacement of the information
1740 in a Resource in conjunction with the interface that is also applied to the operation. The UPDATE
1741 operation is initiated by the Client and consists of three steps, as depicted in Figure 18 and
1742 described below.



1743

1744

Figure 18. UPDATE operation

1745 8.4.1 UPDATE request

1746 The UPDATE request message is transmitted by the Client to the Server to request the update of
1747 information of a Resource on the Server. The UPDATE request message will carry the following
1748 parameters.

- 1749 • *fr*: Unique identifier of the Client
- 1750 • *to*: URI of the resource targeted for the information update
- 1751 • *ri*: Identifier of the UPDATE request
- 1752 • *op*: UPDATE
- 1753 • *cn*: Information, including properties, of the resource to be updated at the target resource

1754 8.4.2 Processing by the Server

1755 Following the receipt of an UPDATE request, the Server may validate if the Client has the
1756 appropriate rights for updating the requested data. If the validation is successful the Server
1757 updates the target Resource information according to the information carried in *cn* parameter of
1758 the UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE
1759 request for use in the response.

1760 An UPDATE request that includes Properties that are read-only shall be rejected by the Server
1761 with an *rs* indicating a bad request.

1762 An UPDATE request shall be applied only to the Properties in the target resource visible via the
1763 applied interface that support the operation. An UPDATE of non-existent Properties is ignored.

1764 8.4.3 UPDATE response

1765 The UPDATE response message will include the following parameters:

- 1766 • *fr*: Unique identifier of the Server
- 1767 • *to*: Unique identifier of the Client
- 1768 • *ri*: Identifier included in the UPDATE request

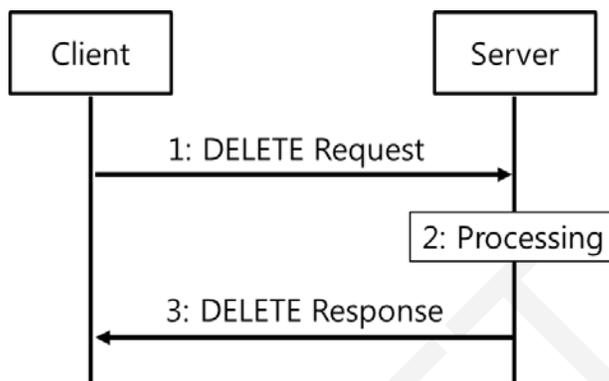
1769 • *rs*: The result of the UPDATE request

1770 The UPDATE response message may also include the following parameters:

1771 • *cn*: The Resource representation following processing of the UPDATE request

1772 8.5 DELETE

1773 The DELETE operation is used to request the removal of a Resource. The DELETE operation is
1774 initiated by the Client and consists of three steps, as depicted in Figure 19 and described below.



1775

1776 **Figure 19. DELETE operation**

1777 8.5.1 DELETE request

1778 DELETE request message is transmitted by the Client to the Server to delete a Resource on the
1779 Server. The DELETE request message will carry the following parameters:

1780 • *fr*: Unique identifier of the Client

1781 • *to*: URI of the target resource which is the target of deletion

1782 • *ri*: Identifier of the DELETE request

1783 • *op*: DELETE

1784 8.5.2 Processing by the Server

1785 Following the receipt of a DELETE request, the Server may validate if the Client has the
1786 appropriate rights for deleting the identified resource, and whether the identified resource exists.
1787 If the validation is successful, the Server removes the requested resource and deletes all the
1788 associated information. The Server caches the value of *ri* parameter in the DELETE request for
1789 use in the response.

1790 8.5.3 DELETE response

1791 The Server shall transmit a DELETE response message in response to a DELETE request
1792 message from a Client. The DELETE response message will include the following parameters.

1793 • *fr*: Unique identifier of the Server

1794 • *to*: Unique identifier of the Client

1795 • *ri*: Identifier included in the DELETE request

1796 • *rs*: The result of the DELETE operation

1797 **8.6 NOTIFY**

1798 The NOTIFY operation is used to request asynchronous notification of state changes. Complete
1799 description of the NOTIFY operation is provided in section 11.4. The NOTIFY operation uses the
1800 NOTIFICATION response message which is defined here.

1801 **8.6.1.1 NOTIFICATION response**

1802 The NOTIFICATION response message is sent by a Server to notify the URLs identified by the
1803 Client of a state change. The NOTIFICATION response message carries the following parameters.

- 1804 • *fr*: Unique identifier of the Server
- 1805 • *to*: URI of the Resource target of the NOTIFICATION message
- 1806 • *ri*: Identifier included in the CREATE request
- 1807 • *op*: NOTIFY
- 1808 • *cn*: The updated state of the resource

1809 **9 Network and connectivity**

1810 **9.1 Introduction**

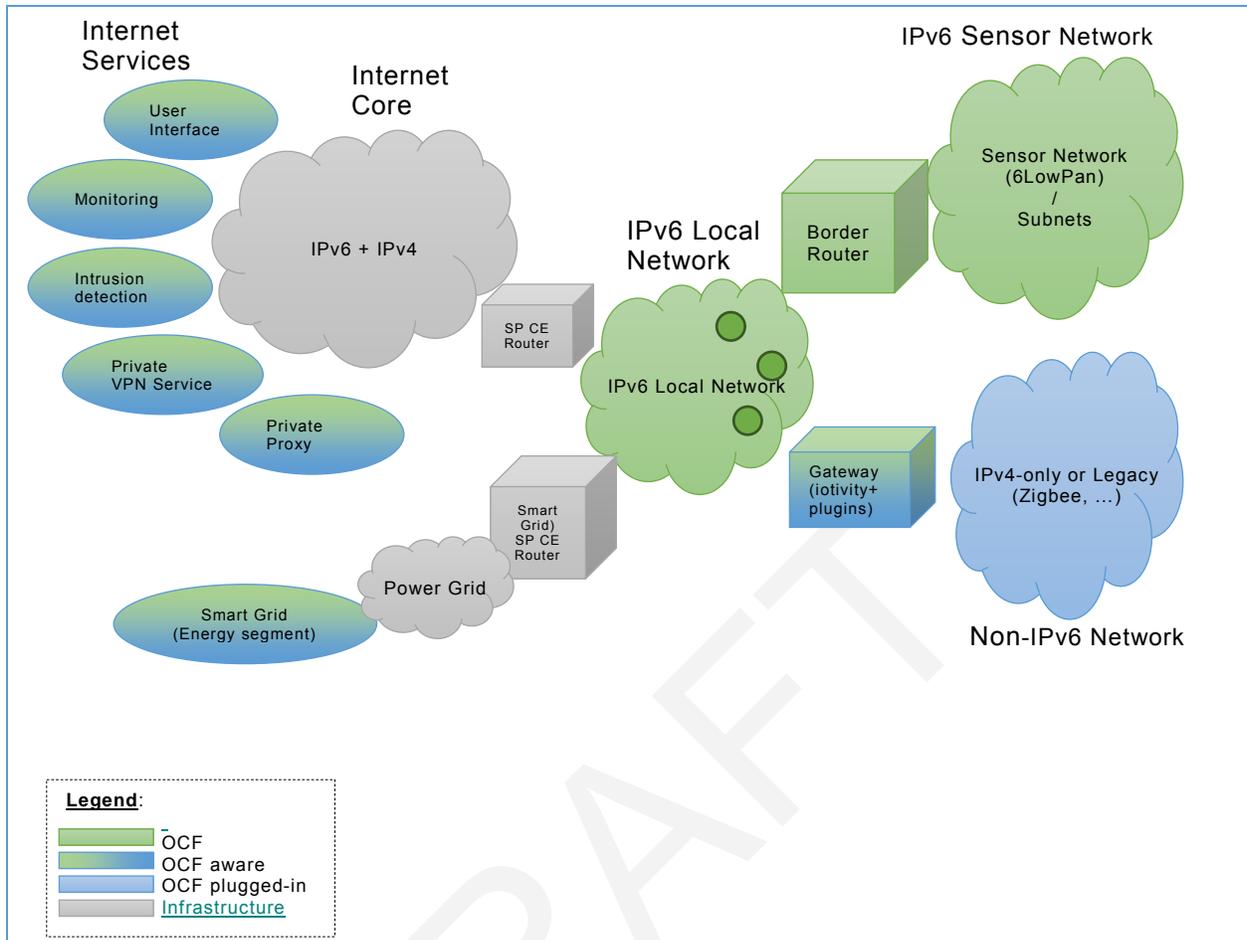
1811 The Internet of Things is comprised of a wide range of applications which sense and actuate the
1812 physical world with a broad spectrum of device and network capabilities: from battery powered
1813 nodes transmitting 100 bytes per day and able to last 10 years on a coin cell battery, to mains
1814 powered nodes able to maintain MBit video streams. It is estimated that many 10s of billions of
1815 IoT devices will be deployed over the coming years.

1816

1817 **9.2 It is desirable that the connectivity options be adapted to the IP layer. To that end,**
1818 **IETF has completed considerable work to adapt Bluetooth®, Wi-Fi, 802.15.4, LPWAN,**
1819 **etc. to IPv6. These adaptations, plus the larger address space and improved**
1820 **address management capabilities, make IPv6 the clear choice for the OCF network**
1821 **layer technology.**

1822 While the aging IPv4 centric network has evolved to support complex topologies, its deployment
1823 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More
1824 complex network topologies, often seen in residential home, are mostly introduced through the
1825 acquisition of additional home network devices, which rely on technologies like private Network
1826 Address Translation (NAT). These technologies require expert assistance to set up correctly and
1827 should be avoided in a home network as they most often result in breakage of constructs like
1828 routing, naming and discovery services.

1829 The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices
1830 and associated routers, but also new services introducing additional edge routers. All these new
1831 requirements require advance architectural constructs to address complex network topologies like
1832 the one shown in Figure 20.



1833

1834

Figure 20. High Level Network & Connectivity Architecture

1835 In terms of RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further
 1836 implement various specializations of those roles. In terms of RFC 6434, IPv6 nodes assume either a router
 1837 or host role. Nodes may further implement various specializations of those roles:

- 1838 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
- 1839 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity
 1840 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).
 1841 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU
 1842 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),
- 1843 • A node may translate and route messaging between IPv6 and non-IPv6 networks.
- 1844 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
- 1845 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity
 1846 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).
 1847 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU
 1848 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),

1849 **9.3 • A node may translate and route messaging between IPv6 and non-IPv6**
1850 **networks.IPv6 network layer requirements**

1851 **9.3.1 Introduction**

1852 Projections indicate that many 10s of billions of new IoT endpoints and related services will be
1853 brought online in the next few years. These endpoint's capabilities will span from battery powered
1854 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating
1855 over Ethernet and WiFi links.

1856 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide
1857 variety of applications such as Web browsing, email, voice, video, and critical system monitoring
1858 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which
1859 is that available address space has been consumed.

1860 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF
1861 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 1862 • Larger address space. Side-effect: greatly reduce the need for NATs.
- 1863 • More flexible addressing architecture. Multiple addresses and types per interface: Link-local,
1864 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed
1865 networks, better re-numbering capability, etc.
- 1866 • More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 1867 • Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 1868 • All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
- 1869 • Major Service Providers around the globe are deploying IPv6.

1870 **9.3.2 IPv6 node requirements**

1871 **9.3.2.1 Introduction**

1872 In order to ensure network layer services interoperability from node to node, mandating a common
1873 network layer across all nodes is vital. The protocol should enable the network to be: secure,
1874 manageable, scalable and to include constrained and self-organizing meshed nodes. OCF
1875 mandates IPv6 as the common network layer protocol to ensure interoperability across all Devices.
1876 More capable devices may also include additional protocols creating multiple-stack devices. The
1877 remainder of this section will focus on interoperability requirements for IPv6 hosts, IPv6
1878 constrained hosts and IPv6 routers. The various protocol translation permutations included in
1879 multi-stack gateway devices may be addresses in subsequent addendums of this specification.

1880 **9.3.2.2 IP Layer**

1881 An IPv6 node shall support IPv6 and it shall conform to the requirements as specified in
1882 IETF RFC 6434:

1883 .

1884 **10 Endpoint**

1885 **10.1 Endpoint definition**

1886 An (OCF) Endpoint is defined as the source or destination of a request and response messages
1887 for a given Transport Protocol Suites (e.g. CoAP over UDP over IPv6). The specific definition of
1888 an Endpoint depends on the Transport Protocol Suites being used. For the example of CoAP over
1889 UDP over IPv6, the endpoint is identified by an IPv6 address and UDP port number.

1890 Each OCF Device shall associate with at least one Endpoint with which it can exchange request
1891 and response messages. When a message is sent to an Endpoint, it shall be delivered to the OCF

1892 Device which is associated with the Endpoint. When a request message is delivered to an Endpoint,
1893 path component is enough to locate the target Resource.

1894 OCF Device can be associated with multiple Endpoints. For example, an OCF Device can have
1895 several IP addresses or port numbers or support both CoAP and HTTP transfer protocol.

1896 On the other hand, an Endpoint can be shared among multiple OCF Devices, only when there is a
1897 way to clearly designate the target Resource with request URI. For example, when multiple CoAP
1898 servers use uniquely different URI paths for all their hosted Resources, and the CoAP
1899 implementation demuxes by path, they can share the same CoAP Endpoint. However, this is not
1900 possible for OIC 1.1 and OCF 1.0 because pre-determined URI (e.g. /oic/d) is mandatory for some
1901 mandatory Resources (e.g. "oic.wk.d").

1902 10.2 Endpoint information

1903 10.2.1 Introduction

1904 Endpoint is represented by Endpoint information which consists of two items of key-value pair,
1905 "ep" and "pri".

1906 10.2.2 "ep"

1907 "ep" represents Transport Protocol Suite and Endpoint Locator specified as follows:

- 1908 • **Transport Protocol Suites** - a combination of protocols (e.g. CoAP + UDP + IPv6) with which
1909 request and response messages can be exchanged for RESTful transaction (i.e. CRUDN).
1910 Transport Protocol Suites shall be indicated by IANA registered schemes (e.g. "coap" or
1911 "coaps" in Table X). Vendor or OCF defined schemes are also allowed (e.g. "org.ocf.foo" or
1912 "com.samsung.bar").
- 1913 • **Endpoint Locator** – an address (e.g. IPv6 address + Port number) through which a message
1914 can be sent to the Endpoint and in turn associated OCF Device. The Endpoint Locator for
1915 "coap", "coaps", "coap+tcp", "coaps+tcp", "http", and "https" shall be specified as "IP address:
1916 port number". Temporary addresses should not be used because Endpoint Locators are for the
1917 purpose of accepting incoming sessions, whereas temporary addresses are for initiating
1918 outgoing sessions (IETF RFC 4941). Moreover its inclusion in /oic/res can cause a privacy
1919 concern (IETF RFC 7721).

1920 "ep" shall have as its value a URI (as specified in IETF RFC 3986) with the scheme component
1921 indicating Transport Protocol Suites and the authority component indicating the Endpoint Locator.
1922 Figure 21 illustrate an exmaple.

1923

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

1924

Figure 21: Example of "ep"

1925 The current list of "ep" with corresponding Transport Protocol Suites is shown in Table 12:

1926

Table 12. "ep" value for Transport Protocol Suites

Transport Protocol Suites	scheme	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	coap+tcp://[2001:db8:a::123]:2222

coaps + tcp + ip	coaps+tcp	IP address + port number	coaps+tcp://[2001:db8:a::123]:2233
http + tcp + ip	http	IP address + port number	http://[2001:db8:a::123]:1111
https + tcp + ip	https	IP address + port number	https://[2001:db8:a::123]:1122

1927 **10.2.3 “pri”**

1928 When there are multiple Endpoints, "pri" indicates the priority among them.

1929 "pri" shall be represented as a positive integer (e.g. "pri": 1) and the lower the value, the higher
1930 the priority.

1931 The default "pri" value is 1, i.e. when "pri" is not present, it shall be equivalent to "pri": 1.

1932 **10.2.4 Endpoint information in "eps" Parameter**

1933 To carry Endpoint information, a new Link Parameter "eps" is defined in 7.8.2.1.6. "eps" has an
1934 array of items as its value and each item represents Endpoint information with two key-value pairs,
1935 "ep" and "pri", of which "ep" is mandatory and "pri" is optional. Figure 22 illustrates a link with
1936 "eps".

1937

```

{
  "anchor": "ocf://light_device_id",
  "href": "/myLightSwitch",
  "rt": ["oic.r.switch.binary"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "p": {"bm": 3},
  "eps": [{"ep": "coap://[fe80::b1d6]:1111", "pri": 2}, {"ep":
"coaps://[fe80::b1d6]:1122"}]
}

```

Figure 22: Example of Link with "eps" Parameter

1938

1939 In Figure 22, "anchor" represents the hosting OCF Device, "href", target Resource and "eps" the
1940 two Endpoints for the target Resource.

1941 If the target Resource of a Link requires a secure connection (e.g. CoAPS), "eps" Parameter shall
1942 be used to indicate the necessary information (e.g. port number) in OCF 1.0 payload, because
1943 "sec" and "port" shall be used only in OIC 1.1 payload.

1944 **10.3 Endpoint discovery**

1945 **10.3.1 Introduction**

1946 "Endpoint discovery" is defined as the process for a Client to acquire the Endpoint information for
1947 OCF Device or Resource.

1948 **10.3.2 Implicit discovery**

1949 If a Device is the source of a CoAP message (e.g. /oic/res response), the source IP address and
1950 port number can be combined to form the Endpoint Locator for the Device. Along with a "coap"
1951 scheme and default "pri" value, Endpoint information for the Device can be constructed.

1952 In other words, an /oic/res response message with CoAP can implicitly carry the Endpoint
1953 information of the responding Device and in turn all the hosted Resources, which can be accessed
1954 with the same transfer protocol of CoAP.

1955 **10.3.3 Explicit discovery with /oic/res response**

1956 Endpoint information can be explicitly indicated with the "eps" Parameter of the Links in /oic/res.

1957 As in 10.3.2, an /oic/res response can implicitly indicate the Endpoint information for the target
1958 Resources hosted by the responding Device. However /oic/res may expose a target Resource
1959 which belongs to another Device. When the Endpoint for a target Resource of a Link cannot be
1960 implicitly inferred, the "eps" Parameter shall be included to provide explicit Endpoint information
1961 with which a Client can access the target Resource.

1962 This applies to the case of /oic/res for a Resource Directory or Bridge Device which usually carries
1963 the Links for Resources which another Device hosts.

1964 Figure 23 is a /oic/res response with the "eps" Parameter in Links.

1965

```
[
  {
    "href": "ocf://bridge_device_id/oic/res/",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "rel": "self",
    {
      "href": "/oic/d",
      "rt": ["oic.d.bridge", "oic.wk.d"],
      "if": ["oic.if.r", "oic.if.baseline"],
      "p": {"bm": 3}},
    {
      "href": "/oic/p",
      "rt": ["oic.wk.p"],
      "if": ["oic.if.r", "oic.if.baseline"],
      "p": {"bm": 3}},

    {
      "anchor": "ocf://light_device_id",
      "href": "/oic/res",
      "rt": ["oic.wk.res"],
      "if": ["oic.if.ll", "oic.if.baseline"],
      "p": {"bm": 3},
      "eps": [{"ep": "coap://[[2001:db8:a::123]:2222}]]},
    {
      "anchor": "ocf://light_device_id",
      "href": "/oic/d",
      "rt": ["oic.d.light", "oic.wk.d"],
      "if": ["oic.if.r", "oic.if.baseline"],
      "p": {"bm": 3},
      "eps": [{"ep": "coap://[[2001:db8:a::123]:2222}]]},
    {
      "anchor": "ocf://light_device_id",
      "href": "/oic/p",
      "rt": ["oic.wk.p"],
      "if": ["oic.if.r", "oic.if.baseline"],
      "p": {"bm": 3},
      "eps": [{"ep": "coap://[[2001:db8:a::123]:2222}]]},
    {
      "anchor": "ocf://light_device_id",
      "href": "/myLightSwitch",
      "rt": ["oic.r.switch.binary"],
      "if": ["oic.if.a", "oic.if.baseline"],
      "p": {"bm": 3},
      "eps": [{"ep": "coap://[[2001:db8:a::123]:2222}]]},

    {
      "anchor": "ocf://fan_device_id",
      "href": "/oic/res",
      "rt": ["oic.wk.res"],
      "if": ["oic.if.ll", "oic.if.baseline"],
```

```

    "p": {"bm":3},
    "eps": [{"ep": "coap://[2001:db8:a::123]:3333"}],
  {
    "anchor": "ocf://fan_device_id",
    "href": "/oic/d",
    "rt": ["oic.d.fan", "oic.wk.d"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm":3},
    "eps": [{"ep": "coap://[2001:db8:a::123]:3333"}],
  {
    "anchor": "ocf://fan_device_id",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm":3},
    "eps": [{"ep": "coap://[2001:db8:a::123]:3333"}],
  {
    "anchor": "ocf://fan_device_id",
    "href": "/myFanSwitch",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],
    "p": {"bm":3},
    "eps": [{"ep": "coap://[2001:db8:a::123]:3333"}]}
  ]

```

Figure 23: Example of /oic/res with Endpoint information

1966

1967 The exact format of the /oic/res response and a way for a Client to acquire a /oic/res response
 1968 message is specified in D.10 and 11.3.5 respectively.

1969 **10.4 CoAP based Endpoint discovery**

1970 The following describes CoAP based Endpoint discovery:

- 1971 a) Advertising or publishing Devices shall join the ‘All OCF Nodes’ multicast groups (as defined
 1972 in [IANA IPv6 Multicast Address Space Registry]) with scopes 2, 3, and 5 (i.e, ff02::158,
 1973 ff03::158 and ff05::158) and shall listen on the port 5683. For compliance to IETF RFC 7252 a
 1974 Device may additionally join the ‘All CoAP Nodes’ multicast groups.
- 1975 b) Clients intending to discover resources shall join the multicast groups as defined in a).
- 1976 c) Clients shall send discovery requests (GET request) to the ‘All OCF Nodes’ multicast group
 1977 address with scope 2 (ff02::158) at port 5683. The requested URI shall be /oic/res. For
 1978 compliance to IETF RFC 7252 a Client may additionally send to the ‘All CoAP Nodes’ multicast
 1979 groups.
- 1980 d) If the discovery request is intended for a specific Resource Type, the Query parameter "rt" shall
 1981 be included in the request (section 6.2.1) with its value set to the desired Resource Type. Only
 1982 Devices hosting the Resource Type shall respond to the discovery request.
- 1983 e) When the “rt” Query parameter is omitted, all Devices shall respond to the discovery request.
- 1984 f) Handling of multicast requests shall be as described in section 8 of IETF RFC 7252 and section
 1985 4.1 in IETF RFC 6690.
- 1986 g) Devices which receive the request shall respond using CBOR payload encoding. A Device shall
 1987 indicate support for CBOR payload encoding for multicast discovery as described in section
 1988 12.4. Later versions of the specification may support alternate payload encodings (JSON,
 1989 XML/EXI, etc.).
- 1990 h)
- 1991

1992 **11 Functional interactions**

1993 **11.1 Introduction**

1994 The functional interactions between a Client and n Server are described in section 11.2 through
 1995 section 11.6 respectively. The functional interactions use CRUDN messages (section 8) and
 1996 include Discovery, Notification, and Device management. These functions require support of core
 1997 defined resources as defined in Table 13. More details about these resources are provided later
 1998 in this section.

1999 **Table 13. 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
/oic/con	Configuration	oic.wk.con	Device Management	No
/oic/mnt	Maintenance	oic.wk.mnt	Device Management	No

2000

2001 **11.2 Onboarding, Provisioning, and Configuration**

2002 Onboarding and Provisioning are fully defined by the OCF Security Specification.

2003
 2004 Should a Device support Client update of configurable information it shall do so via exposing the
 2005 Core Resource /example/oic/con (Table 14) in /oic/res;

2006

2007 **Table 14. Configuration Resource**

Example URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/example/oic/con	Device Configuration	oic.wk.con	oic.if.rw	The Resource Type through which configurable information specific to the Device is exposed. The resource properties exposed in oic.wk.con are listed in Table 15.	Configuration
/example/oic/con	Platform Configuration	oic.wk.con.p	oic.if.rw	The optional Resource Type through which configurable information specific to the Platform is exposed. The resource properties exposed in oic.wk.con.p are listed in Table 16.	Configuration

2008

2009 Table 15 defines the oic.wk.con resource type.

2010

Table 15. oic.wk.con Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n (Common Property of /example/oic/con)	string			R, W	yes	Human friendly name configurable by the end user (e.g. Bob's thermostat). "n" Common Property of /example/oic/con and "n" Common Property /oic/d shall have the same Value. When the "n" Common Property Value of /example/oic/con is modified, it shall be reflected to the "n" Common Property of /oic/d.
Location	loc	array of float (has two elements, the first is latitude, the second is longitude)		Degrees	R, W	no	Provides location information where available.
Location Name	locn	string			R, W	no	Human friendly name for location For example, "Living Room".
Currency	c	string			R,W	no	Indicates the currency that is used for any monetary transactions
Region	r	string			R,W	no	Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote (").
Localized Names	ln	array			R,W	no	Human-friendly name of the Device, in one or more languages. This property is an array of objects where each object has a 'language' field (containing an IETF RFC 5646 language tag) and a 'value' field containing the device name in the indicated language. If this property and the Device Name (n) property are both supported, the Device Name (n) value shall be included in this array.
Default Language	dl	language-tag			R,W	no	The default language supported by the Device, specified as an IETF RFC 5646 language tag. By default, clients can treat any string

							property as being in this language unless the property specifies otherwise.
--	--	--	--	--	--	--	---

2012

2013 Table 16 defines the oic.wk.con.p resource type.

2014

Table 16. oic.wk.con.p Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform Names	mnpn	array			R,W	no	<p>Friendly name of the Platform. 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 platform friendly name in the indicated language.</p> <p>For example, <pre>[{"language": "en", "value": "Dave's Laptop"}]</pre></p>

2015

2016

2017 11.3 Resource discovery

2018 11.3.1 Introduction

2019 Discovery is a function which enables endpoint discovery as well as resource based discovery.
 2020 Endpoint discovery is described in detail in section 10. This section mainly describes the resource
 2021 based discovery.

2022 11.3.2 Resource based discovery: mechanisms

2023 11.3.2.1 Overview

2024 As part of discovery, a Client may find appropriate information about other OCF peers. This
 2025 information could be instances of Resources, Resource Types or any other information
 2026 represented in the resource model that an OCF peer would want another OCF peer to discover.

2027 At the minimum, Resource based discovery uses the following:

- 2028 1) A resource to enable discovery shall be defined. The representation of that resource shall
 2029 contain the information that can be discovered.
- 2030 2) The resource to enable discovery shall be specified and commonly known a-priori. A Device for
 2031 hosting the resource to enable discovery shall be identified.
- 2032 3) A mechanism and process to publish the information that needs to be discovered with the
 2033 resource to enable discovery.
- 2034 4) A mechanism and process to access and obtain the information from the resource to enable
 2035 discovery. A query may be used in the request to limit the returned information.
- 2036 5) A scope for the publication
- 2037 6) A scope for the access.
- 2038 7) A policy for visibility of the information.

2039

2040 Depending on the choice of the base aspects defined above, the Framework defines three resource
2041 based discovery mechanisms:

- 2042 • Direct discovery, where the Resources are published locally at the Device hosting the
2043 resources and are discovered through peer inquiry.
- 2044 • Indirect discovery, where Resources are published at a third party assisting with the
2045 discovery and peers publish and perform discovery against the resource to enable
2046 discovery on the assisting 3rd party.
- 2047 • Advertisement discovery, where the resource to enable discovery is hosted local to the
2048 initiator of the discovery inquiry but remote to the Devices that are publishing discovery
2049 information.

2050 A Device shall support direct discovery.

2051 **11.3.2.2 Direct discovery**

2052 In direct discovery,

- 2053 1) The Device that is providing the information shall host the resource to enable discovery.
- 2054 2) The Device publishes the information available for discovery with the local resource to
2055 enable discovery (i.e. local scope).
- 2056 3) Clients interested in discovering information about this Device shall issue RETRIEVE
2057 requests directly to the resource. The request may be made as a unicast or multicast.
2058 The request may be generic or may be qualified or limited by using appropriate queries in
2059 the request.
- 2060 4) The “server” Device that receives the request shall send a response with the discovered
2061 information directly back to the requesting “client” Device.
- 2062 5) The information that is included in the request is determined by the policies set for the
2063 resource to be discovered locally on the responding Device.
2064

2065 **11.3.2.3 Indirect discovery of Resources (resource directory based discovery)**

2066 In indirect discovery the information about the resource to be discovered is hosted on a Server
2067 that is not hosting the resource. See section 11.3.6 for details on resource directory based
2068 discovery.

2069 In indirect discovery:

- 2070 a) The resource to be discovered is hosted on a Device that is neither the client initiating
2071 the discovery nor the Device that is providing or publishing the information to be
2072 discovered. This Device may use the same resource to provide discovery for multiple
2073 agents looking to discover and for multiple agents with information to be discovered.
- 2074 b) The Device to be discovered or with information to discover, publishes that information
2075 with resource to be discovered on a different Device. The policies on the information
2076 shared including the lifetime/validity are specified by the publishing Device. The
2077 publishing Device may modify these policies as required.
- 2078 c) The client doing the discovery may send a unicast discovery request to the Device
2079 hosting the discovery information or send a multicast request that shall be monitored and
2080 responded to by the Device. In both cases, the Device hosting the discovery information
2081 is acting on behalf of the publishing Device.
- 2082 d) The discovery policies may be set by the Device hosting the discovery information or by
2083 the party that is publishing the information to be discovered. The discovery information

2084 that is returned in the discovery response shall adhere to the policies that are in effect at
2085 the time of the request.
2086

2087 **11.3.2.4 Advertisement Discovery**

2088 In advertisement discovery:

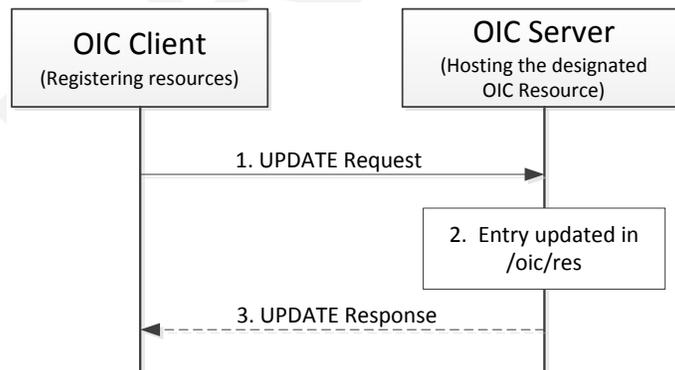
- 2089 a) The resource to enable discovery is hosted local to the Device that is initiating the discovery
2090 request (client). The resource to enable discovery may be an Core Resource or discovered
2091 as part of a bootstrap.
- 2092 b) The request could be an implementation dependent lookup or be a local RETRIEVE request
2093 against the resource that enables discovery.
- 2094 c) The Device with information to be discovered shall publish the appropriate information to
2095 the resource that enables discovery.
- 2096 d) The publishing Device is responsible for the published information. The publishing Device
2097 may UPDATE the information at the resource to enable discovery based on its needs by
2098 sending additional publication requests. The policies on the information that is discovered
2099 including lifetime is determined by the publishing Device.

2100

2101 **11.3.3 Resource based discovery: Information publication process**

2102 The mechanism to publish information with the resource to enable discovery can be done either
2103 locally or remotely. The publication process is depicted in Figure 24. The Device which has
2104 discovery information to publish shall a) either update the resource that enables discovery if
2105 hosted locally or b) issue an UPDATE request with the information to the Device which hosts the
2106 resource that enables discovery. The Device hosting the resource to enable discovery
2107 adds/updates the resource to enable discovery with the provided information and then responds
2108 to the Device which has requested the publication of the resource with an UPDATE response.

2109



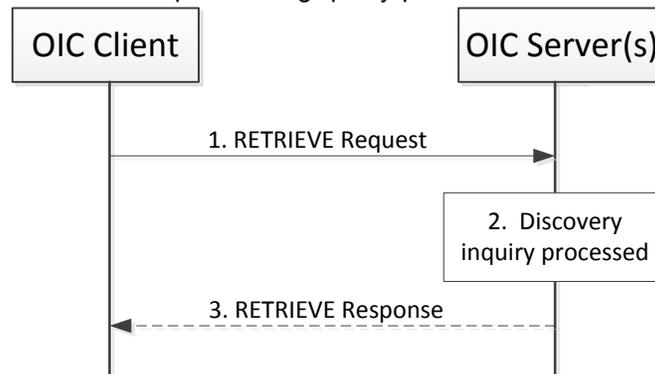
2110

2111 **Figure 24. Resource based discovery: Information publication process**

2112 **11.3.4 Resource based discovery: Finding information**

2113 The discovery process (Figure 25) is initiated as a RETRIEVE request to the resource to enable
2114 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as
2115 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the
2116 support in the data connectivity layer. The response to the request has the information to be
2117 discovered based on the policies for that information. The policies can determine which information
2118 is shared, when and to which requesting agent. The information that can be discovered can be
2119 resources, types, configuration and many other standards or custom aspects depending on the

2120 request to appropriate resource and the form of request. Optionally the requester may narrow the
2121 information to be returned in the request using query parameters in the URI query.



2122

2123

Figure 25. Resource based discovery: Finding information

2124

2125 **Discovery Resources**

2126 Some of the Core Resources shall be implemented on all Devices to support discovery. The Core
2127 Resources that shall be implemented to support discovery are:

- 2128 ● /oic/res for discovery of resources
- 2129 ● /oic/p for discovery of platform
- 2130 ● /oic/d for discovery of device information

2131 Details for these mandatory Core Resources are described in Table 17

2132 Platform resource –

2133 The OCF recognizes that more than one instance of Device may be hosted on a single platform.
2134 Clients need a way to discover and access the information on the platform. The core resource,
2135 /oic/p exposes platform specific properties. All instances of Device on the same Platform shall
2136 have the same values of any properties exposed (i.e. a Device may choose to expose optional
2137 properties within /oic/p but when exposed the value of that property should be the same as the
2138 value of that property on all other Devices on that Platform)

2139

2140 Device resource

2141 The device resource shall have the pre-defined URI /oic/d. The resource /oic/d exposes the
2142 properties pertaining to a Device as defined in Table 17. The properties exposed are determined
2143 by the specific instance of Device and defined by the Resource Type(s) of /oic/d on that Device.
2144 Since all the Resource Types of /oic/d are not known a priori, the Resource Type(s) of /oic/d shall
2145 be determined by discovery through the core resource /oic/res. The device resource /oic/d shall
2146 have a default Resource Type that helps in bootstrapping the interactions with this device (the
2147 default type is described in Table 17.)

2148

2149 Protocol indication

2150 A Device may need to support different messaging protocols depending on requirements for
2151 different application profiles. For example, the Smart Home profile may use CoAP and the
2152 Industrial profile may use DDS. To enable interoperability, a Device uses the protocol indication
2153 to indicate the transport protocols they support and can communicate over.

2154

Table 17. Mandatory discovery Core Resources

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/res	Default	oic.wk.res	oic.if.ll	<p>The resource through which the corresponding Server is discovered and introspected for available resources.</p> <p>/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 link 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 resource properties exposed by /oic/res are listed in Table 18.</p>	Discovery
/oic/p	Platform	oic.wk.p	oic.if.r	<p>The discoverable resource through which platform specific information is discovered.</p> <p>The resource properties exposed by /oic/p are listed in Table 20</p>	Discovery
/oic/d	Device	oic.wk.d and/or one Device Specific Resource Type ID	oic.if.r	<p>The discoverable via /oic/res resource which exposes properties specific to the Device instance.</p> <p>The resource properties exposed by /oic/d are listed in Table 20</p> <p>/oic/d may have one Resource Type that is specific to the Device in addition to the default Resource Type or if present overriding the default Resource Type.</p> <p>The base type oic.wk.d defines the properties that shall be exposed by all Devices.</p> <p>The device specific Resource Type exposed is dependent on the class of device (e.g. air conditioner, smoke alarm); applicable values are defined by the vertical specifications.</p>	Discovery

2156

2157 Table 18 defines oic.wk.res Resource Type.

2158

Table 18. oic.wk.res Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R	no	Human-friendly name defined by the vendor
Links	links	array	See 7.8.2		R	yes	The array of Links describes the URI, supported Resource Types and interfaces, and access policy.
Messaging Protocol	mpro	SSV			R	No	String with Space Separated Values (SSV) of messaging protocols supported as a SI Number from Table 19 For example, "1 and 3" indicates that the Device supports coap and http as messaging protocols.

2159 A Device which wants to indicate its messaging protocol capabilities may add the property 'mpro'
 2160 in response to a request on /oic/res. A Device shall support CoAP based discovery as the baseline
 2161 discovery mechanism (see section 10.4). A Client which sees this property in a discovery response

2162 can choose any of the supported messaging protocols for communicating with the Server for further
 2163 messages. For example, if a Device supporting multiple protocols indicates it supports a value of
 2164 “1 3” for the ‘mpro’ property in the discovery response, then it cannot be assumed that there is an
 2165 implied ordering or priority. But a vertical service specification may choose to specify an implied
 2166 ordering or priority. If the ‘mpro’ property is not present in the response, A Client shall use the
 2167 default messaging protocol as specified in the vertical specification for further communication.

2168 The “/oic/res” shall list all Resources that are indicated as discoverable (see section 11.3). Also
 2169 the following architecture Resource Types shall be listed:

- 2170 • Introspection resource indicated with an “rt” value of “oic.wk.introspection”
- 2171 • /oic/p indicated with an “rt” value of “oic.wk.p”
- 2172 • /oic/d indicated with an “rt” value of “oic.wk.d”
- 2173 • /oic/sec/doxm indicated with an “rt” value of “oic.r.doxm”
- 2174 • /oic/sec/pstat indicated with an “rt” value of “oic.r.pstat”

2175 Conditionally required:

- 2176 • “/oic/res” with an “rt” value of “oic.wk.res” as self-reference, on the condition that “oic/res” has
 2177 to signal that it is observable by a Client.

2178 The Introspection Resource is only applicable for Devices that host vertical Resource Types (e.g.
 2179 “oic.r.switch.binary”) or vendor-defined Resources. Devices that only host Resources required to
 2180 onboard the Device as a Client do not have to implement the Introspection Resource.

2181 Table 19 provides an OCF registry for protocol schemes.

2182 **Table 19. Protocol scheme registry**

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

2183 Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by a Client to form
 2184 requests in a different messaging protocol other than discovery is out of scope.

2185

2186 The following applies to the use of /oic/d as defined above:

- 2187 • A vertical may choose to expose its Device Type (e.g., refrigerator or A/C) by adding the Device
 2188 Type to the list of Resource Types associated with /oic/d.
 - 2189 ○ For example; rt of /oic/d becomes ["oic.wk.d", "oic.d.<thing>"]; where “oic.d.<thing>”
 2190 is defined in another spec such as the Smart Home vertical.
 - 2191 ○ This implies that the properties exposed by /oic/d are by default the mandatory
 2192 properties in Table 20.
- 2193 • A vertical may choose to extend the list of properties defined by the Resource Type 'oic.wk.d'.
 2194 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The
 2195 mandatory properties defined in Table 20 shall always be present.

2196 Note:

2197 As per existing Core specification definitions the Resource Type ID may be a list of Resource Type IDs; when that is the
 2198 case the default Resource Type ID for /oic/d is the first Resource Type ID listed. So a vertical can list 'oic.d.thing' first.
 2199 This then means a GET /oic/d returns the properties for oic.d.thing and a GET /oic/d?rt=<some rt> returns the properties
 2200 for the rt listed in the query.

2201 Table 20 oic.wk.d Resource Type definition defines the base Resource Type for the /oic/d resource.

2202

2203

Table 20. oic.wk.d Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R	no	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			R	yes	Spec version of the core specification this device is implemented to, The syntax is "ocf.<major>.<minor>.<sub-version>" where <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. This version of the specification the string value shall be "ocf.1.0.0".
Device ID	di	uuid			R	yes	Unique identifier for Device. This value shall be the same value (i.e. mirror) as the doxm.deviceuuid Property as defined in OCF Security. Handling privacy-sensitivity for the "di" Property, refer to section 13.8 in OCF Security.
Data Model Version	dmv	csv			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. This version of the specification one entry in the csv string shall be "ocf.res.1.0.0". Another entry in the csv shall be the vertical(s) being (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.
Protocol Independent ID	piid	UUID			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 Protocol Independent ID value for all the protocols it supports. Handling privacy-sensitivity for the "piid" Property, refer to section 13.8 in OCF Security.
Localized Descriptions	ld	array			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			R	no	Version of the device software.
Manufacturer Name	dmn	array			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			R	no	Model number as designated by manufacturer.

2204

2205 The additional Resource Type(s) of the /oic/d resource are defined by the vertical specification.

2206

2207 Table 21 defines oic.wk.p Resource Type.

2208

2209

Table 21. oic.wk.p Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform ID	pi	string			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 section 13.8 in OCF Security.
Manufacturer Name	mnmn	string			R	yes	Name of manufacturer
Manufacturer Details Link	mnml	uri			R	no	Reference to manufacturer, represented as a URI
Model Number	mnmo	string			R	no	Model number as designated by manufacturer
Date of Manufacture	mnmt	date		Time (<i>show RFC</i>)	R	no	Manufacturing date of Platform.
Platform Version	mnpv	string			R	no	Version of platform – string (defined by manufacturer)
OS Version	mnos	string			R	no	Version of platform resident OS – string (defined by manufacturer)
Hardware Version	mnhw	string			R	no	Version of platform hardware
Firmware version	mnfv	string			R	no	Version of Platform firmware
Support link	mnsi	uri			R	no	URI that points to support information from manufacturer
SystemTime	st	date-time			R	no	Reference time for the Platform.
Vendor ID	vid	string			R	no	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.

2210

2211 Composite Device

2212 A physical device may be modelled as a single device or as a composition of other devices. For
 2213 example a refrigerator may be modelled as a composition, as such part of its definition of may
 2214 include a sub-tending thermostat device which itself may be composed of a sub-tending
 2215 thermometer device.

2216 There may be more than one way to model a server as a composition. One example method would
 2217 be to have Platform which represents the composite device to have more than one instance of a
 2218 Device on the Platform. Each Device instance represents one of the distinct devices in the
 2219 composition. Each instance of Device may itself have or host multiple instances of other resources.

2220 An implementation irrespective of how it is composed shall only expose a single instance of `/oic/d`
2221 with an 'rt' of choice for each logical Server.

2222 Thus, for the above refrigerator example if modeled as a single Server; `/oic/res` would expose
2223 `/oic/d` with a Resource Type name appropriate to a refrigerator. The sub-tending thermostat and
2224 thermometer devices would be exposed simply as instances of a resource with a device
2225 appropriate Resource Type with an associated URI assigned by the implementation; e.g.,
2226 `/MyHost/MyRefrigerator/Thermostat` and `/MyHost/MyRefrigerator/Thermostat/Thermometer`.

2227

2228 **11.3.5 Resource discovery using `/oic/res`**

2229 Discovery using `/oic/res` is the default discovery mechanism that shall be supported by all Devices
2230 as follows:

2231 a) Every Device updates its local `/oic/res` with the resources that are discoverable (see section
2232 7.3.2.2). Every time a new resource is instantiated on the Device and if that resource is
2233 discoverable by a remote Device then that resource is published with the `/oic/res` resource that
2234 is local to the Device (as the instantiated resource).

2235 b) A Device wanting to discover resources or Resource Types on one or more remote Devices
2236 makes a RETRIEVE request to the `/oic/res` on the remote Devices. This request may be sent
2237 multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request
2238 may optionally be restricted using appropriate clauses in the query portion of the request.
2239 Queries may select based on Resource Types, interfaces, or properties.

2240 c) The query applies to the representation of the resources. `/oic/res` is the only resource whose
2241 representation has "rt". So `/oic/res` is the only resource that can be used for Multicast discovery
2242 at the transport protocol layer.

2243 d) The Device receiving the RETRIEVE request responds with a list of resources, the Resource
2244 Type of each of the resources and the interfaces that each resource supports. Additionally
2245 information on the policies active on the resource can also be sent. The policy supported
2246 includes observability and discoverability. (More details below)

2247 e) The receiving Device may do a deeper discovery based on the resources returned in the
2248 request to `/oic/res`.

2249

2250 The information that is returned on discovery against `/oic/res` is at the minimum:

- 2251 • The URI (relative or fully qualified URL) of the resource
- 2252 • The Resource Type(s) of each resource. More than one Resource Type may be returned if the
2253 resource enables more than one type. To access resources of multiple types, the specific
2254 Resource Type that is targeted shall be specified in the request.
- 2255 • The Interfaces supported by that Resource. Multiple interfaces may be returned. To access a
2256 specific interface that interface shall be specified in the request. If the interface is not specified,
2257 then the Default Interface is assumed.

2258 Different `/oic/res` responses are returned according to requesting Clients, which indicate their
2259 preference with Content Format in Accept Option. OCF 1.0 Clients request with the Content Format
2260 of "application/vnd.ocf+cbor", whereas the absence of that Content Format
2261 (i.e. "application/vnd.ocf+cbor") indicates OIC 1.1 Clients.

2262 For OIC 1.1 Clients, `/oic/res` response shall use "sec" and "port" to provide the information for an
2263 encrypted connection.

2264 For OCF 1.0 Clients, /oic/res response only includes the “array of Links to conform to
2265 IETF RFC 6690. Each Link shall use "eps" Parameter to provide the information for an encrypted
2266 connection and carry "anchor" of the value OCF URI where the authority component of <deviceId>
2267 indicates the Device hosting the target Resource.

2268 The JSON schemas for discovery using /oic/res are described in D.10. Also refer to Section 10
2269 (Endpoint Discovery) for details of Multicast discovery using /oic/res on a CoAP transport.

2270 For example, a Light device might return the following to OIC 1.1 clients:

```
2271 [
2272   {
2273     "di": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2274     "links": [
2275       {
2276         "href": "coaps://[fe80::b1d6]:44444/oic/res",
2277         "rel": "self",
2278         "rt": ["oic.wk.res"],
2279         "if": ["oic.if.ll", "oic.if.baseline"],
2280         "p": {"bm": 3}
2281       },
2282       {
2283         "href": "/oic/p",
2284         "rt": ["oic.wk.p"],
2285         "if": ["oic.if.r", "oic.if.baseline"],
2286         "p": {"bm": 3, "sec": true, "port": 11111}
2287       },
2288       {
2289         "href": "/oic/d",
2290         "rt": ["oic.wk.d", "oic.d.light"],
2291         "if": ["oic.if.r", "oic.if.baseline"],
2292         "p": {"bm": 3, "sec": true, "port": 11111}
2293       },
2294       {
2295         "href": "/myLight",
2296         "rt": ["oic.r.switch.binary"],
2297         "if": ["oic.if.a", "oic.if.baseline"],
2298         "p": {"bm": 3, "sec": true, "port": 11111}
2299       }
2300     ]
2301   }
2302 ]
```

2303 The light device might return the following to clients that request with the Content Format of
2304 “application/vnd.ocf+cbor” in Accept Option:

```
2305 [
2306   {
2307     "href": "/oic/res",
2308     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989/oic/res",
2309     "rel": "self",
2310     "rt": ["oic.wk.res"],
2311     "if": ["oic.if.ll", "oic.if.baseline"],
2312     "p": {"bm": 3},
2313     "eps": [{"ep": "coap://[fe80::b1d6]:44444"}]
2314   },
2315   {
2316     "href": "/oic/p",
2317     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2318     "rt": ["oic.wk.p"],
2319     "if": ["oic.if.r", "oic.if.baseline"],
2320     "p": {"bm": 3},

```

```

2321     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2322             {"ep": "coaps://[fe80::b1d6]:11111"}
2323         ],
2324     },
2325     {
2326         "href": "/oic/d",
2327         "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2328         "rt": ["oic.wk.d", "oic.d.light"],
2329         "if": ["oic.if.r", "oic.if.baseline"],
2330         "p": {"bm": 3},
2331         "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2332                 {"ep": "coaps://[fe80::b1d6]:11111"}
2333             ],
2334     },
2335     {
2336         "href": "/myLight",
2337         "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2338         "rt": ["oic.r.switch.binary"],
2339         "if": ["oic.if.a", "oic.if.baseline"],
2340         "p": {"bm": 3},
2341         "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2342                 {"ep": "coaps://[fe80::b1d6]:11111"}
2343             ],
2344     }
2345 ]

```

2346 After performing discovery using /oic/res, Clients may discover additional details about Server by
 2347 performing discovery using /oic/p, /oic/rts etc. If a Client already knows about Server it may
 2348 discover using other resources without going through the discovery of /oic/res.

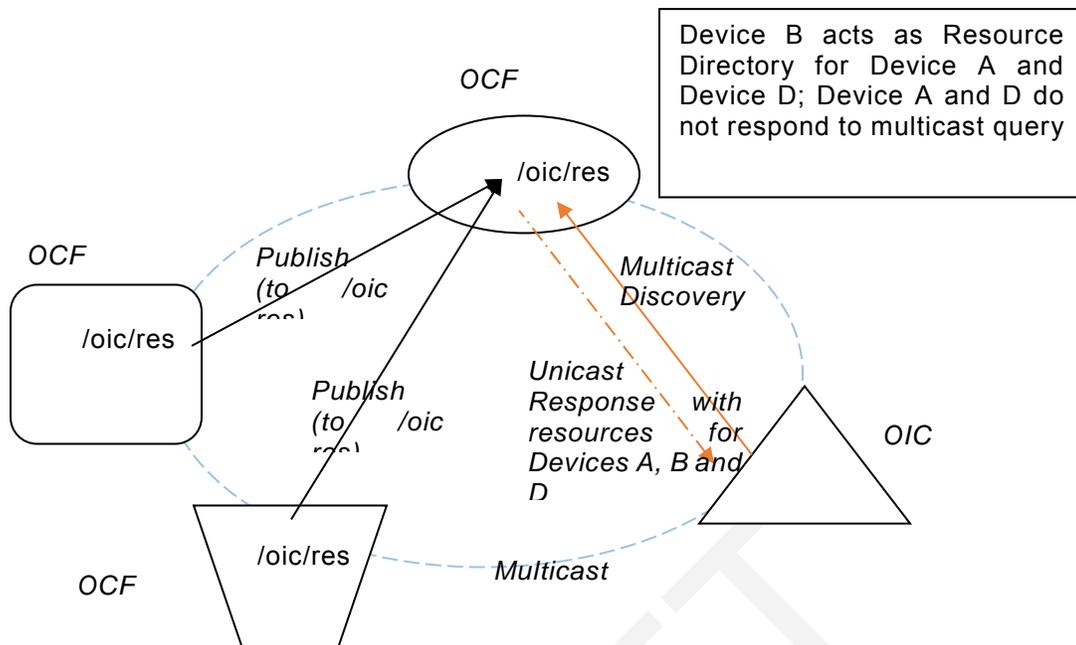
2349 **11.3.6 Resource directory (RD) based discovery**

2350 **11.3.6.1 Introduction**

2351 **11.3.6.1.1 Indirect discovery for lookup of the resources**

2352 Direct discovery is the mechanism used currently to find resources in the network. When needed,
 2353 resources are queried at a particular node directly or a multicast packet is sent to all nodes. Each
 2354 queried node responds directly with its discoverable resources to the discovering device.
 2355 Resources available locally are registered on the same device.

2356 In some situations, one of the other mechanisms described in section 11.3.2.3, called indirect
 2357 discovery, may be required. Indirect discovery is when a 3rd party device, other than the
 2358 discovering device and the discovered device, assists with the discovery process. The 3rd party
 2359 only provides information on resources on behalf of another device but does not host resources
 2360 on part of that device.



2361

2362

Figure 26. Indirect discovery of resource by resource directory

2363 Indirect discovery is useful for a resource constrained device that needs to sleep to manage power
 2364 and cannot process every discovery request, or when devices may not be on the same network
 2365 and requires optimization for discovery. Once resources are discovered using indirect discovery
 2366 then the access to the resource is done by a request directly to the Device that hosts that resource.

2367 **11.3.6.1.2 Resource directory**

2368 A resource directory (RD) is a Device that assists with indirect discovery. A RD can be queried at
 2369 its /oic/res resource to find resources hosted on other Devices. These Devices can be sleepy
 2370 nodes or any other device that cannot or may not respond to discovery requests. Device can
 2371 publish all or partial list of resources they host to a RD. The RD then responds to queries for
 2372 Resource discovery on behalf of the publishing Device (for example: when a Device may go to
 2373 sleep). For general Resource discovery, the RD behaves like any other Server in responding to
 2374 requests to /oic/res.

2375 Any Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The Devices
 2376 that want to discover RDs shall use this resource and one of the Resource discovery mechanisms
 2377 to discover the RD and get the parameters of the RD. The information discovered through this
 2378 resource shall be used to select the appropriate RD to use for resource publication. The bias
 2379 information shall include the following criteria: power source (AC, battery powered or safe/reliable),
 2380 connectivity (wireless, wired), CPU, memory, load statistics (processing publishing and query from
 2381 the devices). In addition, the RD shall return a bias factor that ranges from 0 to 100. Optionally,
 2382 the RD may also return a context - the value which shall be a string and semantics of the context
 2383 are not discussed in this document but it is expected that the context will be used to establish a
 2384 domain, region or some such scope that is meaningful to the application, deployment or usage.

2385 Using these criteria or the bias factor, the Device shall select one RD (per context) to publish its
 2386 resources. A context describes the state of an OCF Device with respect to Resource discovery. A
 2387 context is usually determined at deployment and from application requirements. An example of a
 2388 context could be a multicast group- a Device that is a member of more than one multicast group
 2389 may have to find and select a RD in each of the multicast groups (i.e. per context) to publish its
 2390 information. The Device may choose other RDs during its lifetime but a Device shall not publish

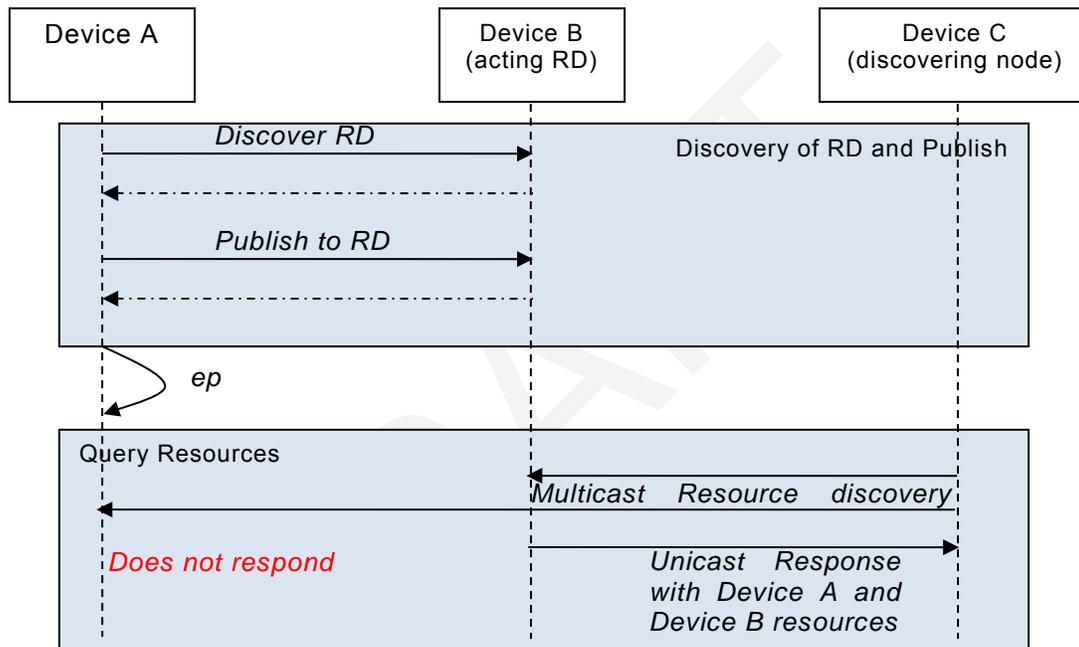
2391 its resource information to more than one RD Devices such as TV, network router, desktop will
2392 have higher weightage or bias factor compared to mobile phone device.

2393 This remainder of this section is divided into two three parts. The first part covers discovering of
2394 the RD. The second part covers and publishing, updating and deleting of resources for the
2395 constrained/sleepy device. Second The third part covers where RD replies to queries from devices
2396 looking to discover resources.

2397 11.3.6.2 Resource directory discovery

2398 11.3.6.2.1 Discovering a resource directory

2399 A RD in the OCF network shall support RD discovery, shall provide the facility to allow devices to
2400 publish their resource information to a RD, to update resource information in a RD and to delete
2401 resource information from a RD.



2402

2403

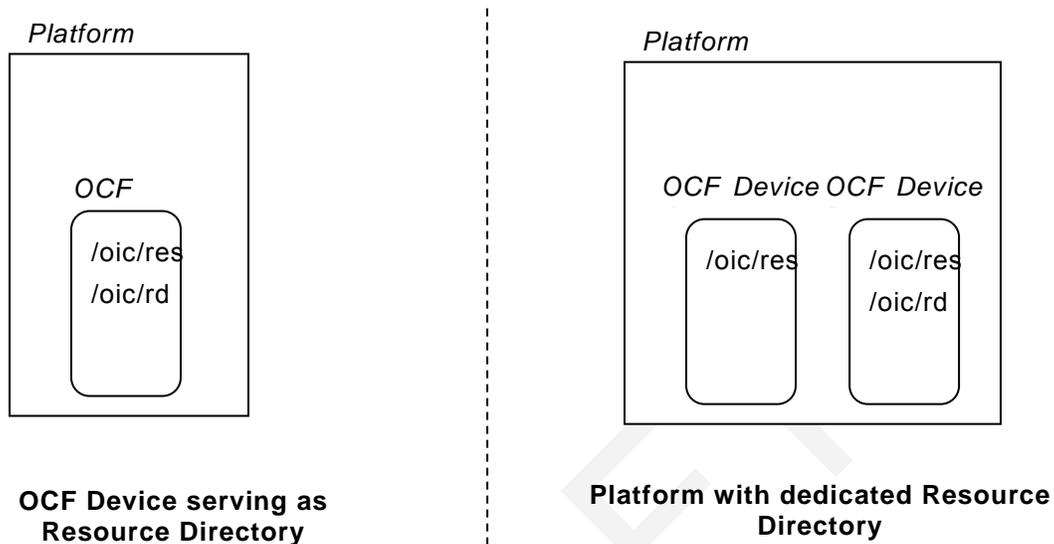
Figure 27. RD discovery and RD supported query of resources support

2404 As shown in Figure 27, the Device that wishes to advertise its resources: first discovers a resource
2405 directory and then publishes the desired resource information. Once a set of resources have been
2406 published to a RD then the publishing device should not respond to multicast Resource discovery
2407 queries for those published resources when the RD is on the same multicast domain. In that case,
2408 only the RD shall respond to multicast Resource discovery requests on the resource published to
2409 it.

2410 An OCF network allows for more than one device acting as a RD. The reason to have multiple RD
2411 support is to make network scalable, handle network failures and centralized device failure
2412 bottleneck. This does not preclude a scenario where a use case or deployment environment may
2413 require single device in the environment to be deployed as the only resource directory (e.g.
2414 gateway model). There may be more than one Device acting as RD on a Platform.

2415 Discovering of an RD may result in responses from more than one RD. The discovering device
2416 shall select a RD. The selection may be based on the weightage parameter(s) provided in the
2417 response from the RD.

2418 An RD will be application agnostic i.e., application should not be aware whether resource directory
 2419 was queried to get the resource information. All the handling of the retrieval is kept opaque to the
 2420 application. A Client that performs Resource discovery uses an RD just like it may use any other
 2421 Server for discovery. It may send a unicast request to the RD when it needs only the resource
 2422 advertised on the RD or do a multicast query when it does not require or have explicit knowledge
 2423 of an RD.



2424
 2425 **Figure 28. Resource Direction Deployment Scenarios**

2426 Resource directory can also be discovered in the following manners:

- 2427 • **Pre-configuration:** Devices wishing to publish resource information may be configured a priori
 2428 with the information (e.g. IP address, port, transport etc.) of a specific resource directory. This
 2429 pre-configuration may be done at onboarding or may be updated on the device using an out-
 2430 of-band method. This pre-configuration may be done by the manufacturer or by the user/device
 2431 manager.
- 2432 • **Query-oriented:** A Client wanting to discover resource directories using query-oriented
 2433 discovery (i.e. pull) shall issue a multicast Resource discovery request for
 2434 /oic/res?rt=oc.wk.rdpub. Only and all Devices that can be a RD will respond to this query. The
 2435 response shall include information about the RD (as defined by the Resource Type) and
 2436 weightage parameters to allow the discovering device to select between RDs (see details in
 2437 RD selection section). The oic.wk.rdpub resource shall be instantiated on the OCF Devices
 2438 acting as a resource directory. The oic.wk.rd.pub schema is as defined in D.14..
- 2439 • **Advertisement:** An RD may advertise about itself to devices. It is an advertisement packet. The
 2440 devices that are already publishing to a RD may use this as a heartbeat message of the RD. If
 2441 the RD advertisement does not arrive at a stipulated interval, publishing device starts searching
 2442 for other RDs in the network, as this is a signal that RD is not online. Other usage of this
 2443 message is it serves as an advertisement for a device seeking a RD to publish their resources.
 2444 The details from the advertisement can then be used to query directly to a RD to get weightage
 2445 details instead of sending a multicast packet in a network. As it is intended this is sent at a
 2446 regular interval and does not include weightage information to keep packet sizes small.
- 2447 • One of the important benefits of an RD is to make services discoverable in networks that don't
 2448 support site wide multicast but do support site wide routing. An example of such a network is
 2449 Homenet .To enable an RD function across such a network a site discovery mechanism is
 2450 needed to discover the RD service (IP address & port number). In order to make itself

2451 discoverable beyond the link local scope, an RD with a routable ip address shall implement the
 2452 mDNS responder requirements defined in IETF RFC 6762. The RD shall respond to mDNS
 2453 queries of type PTR and with a service name equal to "_rd._sub._oic._udp.local". The response
 2454 shall include all routable IP addresses. Devices with a routable ip address shall discover all
 2455 available RD instances by issuing a DNS-SD's PTR lookup as defined in IETF RFC 6763 with
 2456 as service name service name "_rd._sub._oic._udp.local". The response shall include all
 2457 routable addresses/port pair through which the RD service is made accessible.

2458 **11.3.6.2.2 Resource directory selection process**

2459 **11.3.6.2.2.1 Selection criteria**

2460 When a device discovers more than one RD then it shall decide to use one of these RDs based on
 2461 the selection criteria described here. A device shall use or publish information to only one RD
 2462 within a multicast domain at a given time. This is to minimize the burden of processing duplicate
 2463 information in the Resource discovery phase.

2464 There two ways to select an RD. One is based on a bias factor (RD generated) and the other is
 2465 based on clients determination based on granular parameters provided by the server (client/device
 2466 generated). Devices may use one or both methods to select an RD.

2467 *Bias factor:* The bias factor is a server generated positive number in the range of 0 to 100, where
 2468 0 is the lowest to 100 being the highest. If two RDs have the same bias factor then the selecting
 2469 device may choose either based auxiliary criteria or at random. Either way only one RD shall be
 2470 selected and used at a time. No specific method is defined in this specification to determine the
 2471 bias factor for an RD. The number may be a pre-configured value at the time of onboarding or
 2472 subsequent configuration of the RD or may be based on a formula determined by the
 2473 implementation of the RD. (OCF will provide a standard formula for this calculation in a future
 2474 version or release of specification).

2475 The bias factor shall be calculated by the RD by adding the contribution values determined for
 2476 each of the parameters in Table 22 and divided by the number of parameters. An RD may advertise
 2477 a bias factor larger than the calculated value when there is reason to believe that the RD is highly
 2478 capable for example an installed service provider gateway.

2479 *Parameters:* Optionally, parameters defined in Table 22 (like direct power supply, network
 2480 connectivity, load conditions, CPU power, memory, etc.) may be returned in the discovery
 2481 response. Discovering device may use the details to make granular selection decisions based on
 2482 client defined policies and criteria that use the RD parameters. For example, a device in an
 2483 industrial deployment may not weight power connectivity high but another in home environments
 2484 may give more weightage for power.

2485 **Table 22: Selection parameters**

Parameter	Values (Contribution)	Description
Power	Safe (100) AC (70) Batt (40)	<ul style="list-style-type: none"> Safe implies that the power supply is reliable and is backed up with battery for power outages etc. Implementation may lower the number for Batt based on the type of battery the RD device runs on. If battery conservation is important then this number should be lowered.
Mobility	Fixed (100) Mobile (50)	<ul style="list-style-type: none"> Implementation may further grade the mobility number based on how mobile the RD device is; lower number for highly mobile and larger numbers for limited mobility The mobility number shall not be larger than 80
Network Product	Type: <ul style="list-style-type: none"> Wired (10) Wireless (4) 	<ul style="list-style-type: none"> Network product = [sum of (type * bandwidth per network interface)]/[number of interfaces] Normalized to 100

	Bandwidth: <ul style="list-style-type: none"> • High (10) • Low (5) • Lossy (3) Interfaces	
Memory Factor	Available Total	<ul style="list-style-type: none"> • Memory is the volatile or non-volatile storage used to store the resource information • Memory Factor = [Available]/[Total] • Normalized to 100 (i.e. expressed as percentage)
Request Load Factor	1-minute 5-minute 15-minutes	<ul style="list-style-type: none"> • Current request loading of the RD • Similar to UNIX load factor (using observable, pending and processing requests instead of runnable processes) • Expressed as a load factor 3-tuple (up to two decimal points each). Factor is based on request processed in a 1-minute (L1), 5-minute (L5) and 15-minute (L15) windows • See http://www.teamquest.com/import/pdfs/whitepaper/ldavq1.pdf • Factor = $100 - ([L1*3 + L5*7 + L15*10]/3)$

2486

2487 **11.3.6.2.2.2 Selection scenarios**

2488 The device that wants to use an RD will use the endpoint discovery to find zero or more RDs on
2489 the network. After discovering the RDs, the device needs to select an RD of all found RDs on the
2490 network. The selection based on the bias factor will ensure that a Device can judge if the found
2491 RD is suitable for its needs.

2492 The following situation can occur during the selection of an RD:

- 2493 1) A single or multiple RDs are present in the network
- 2494 2) No RD is present in the network
- 2495 3) an additional RD arrives on the network

2496

2497 In the first scenario the RDs are already present. If a single RD is detected then that RD can be
2498 used . When multiple RDs are detected the Device uses the bias information to select the RD.

2499

2500 In the second scenario, device will listen to the advertisement of the devices that hosts the RDs.
2501 Once an RD advertisement packet is received it judges if the bias criteria are met and starts using
2502 the RDs.

2503

2504 In the third scenario the Device has already published its resources to an existing RD. In this
2505 scenario it discovers a new RD on the network.

2506 After judging the bias factor the Device may choose to move to the new RD.

2507

2508 **11.3.6.3 If the decision is made to select the new RD, the then Device shall delete its**
2509 **resource information from the current used RD and then after removal publish**

2510 **the information to the new RD. During the transition period the Device itself**
2511 **shall respond to Resource discovery requests. Resource publishing**

2512 **11.3.6.3.1 Publish resources**

2513 **11.3.6.3.1.1 Overview**

2514 After the selection process of a RD, a device may choose one of the following mechanisms:

- 2515 • Push its resources information to the selected RD or
- 2516 • Request the RD to pull the resource information by doing a unicast discovery request against
2517 its /oic/res

2518 The publishing device may decide to publish all resources or few resources on the resource
2519 directory. The publishing device shall only publish resources that are otherwise published to its
2520 own /oic/res. A publishing device may respond to discovery requests (on its /oic/res resource) for
2521 the resources it does not publish to a RD. Nonetheless, it is highly recommended that when an RD
2522 is used, all discoverable resources on the publisher be published to the RD.

2523 **11.3.6.3.1.2 Publish: Push resource information**

2524 Resource information is published using an UPDATE CRUDN operation to /oic/rd using the
2525 Tesource Type oic.wk.rdpub and the oic.if.baseline interface.

2526 Once a publishing device has published resources to a RD, it should not respond to the multicast
2527 discovery queries for the same resources against its own /oic/res, especially when on the same
2528 multicast domain as the RD. After publishing resources, it is a RD responsibility to reply to the
2529 queries for the published resources.

2530 If the publishing device is in sleep mode and a RD has replied on behalf of the publishing device,
2531 then a discovering device will try to access resource on the provided URI.

2532 There is another possibility that the resource directory and the publishing device both respond to
2533 the multicast query from the discovering device. This will create a duplication of the information
2534 but is an alternate that may be used for non-robust network. It is not a recommended option but
2535 for industrial scenarios, this is one of the possibilities. Either way, discovering clients shall always
2536 be prepared to process duplicate information in responses to multicast discovery request. The
2537 /oic/rd schema is as defined in D.14 to specify publishing (oic.rd.publish) to the /oic/rd resrouce.

2538 **11.3.6.3.2 Update resource information**

2539 Server will hold the publish resource information till the time specified in the ttl field. A device can
2540 send update if it seeks a RD to keep holding resources and reply to queries on its behalf. Update
2541 can be used for updating about all resources that are published on a RD or can use to do per
2542 resource published.

2543 Updates are done using the same Resource Type and interface as for the initial publish but only
2544 the information to be updated is provided in the payload.

2545 **11.3.6.3.3 Delete resource information**

2546 A resource information hold at the resource directory can be removed anytime by the publishing
2547 device. It can be either for the whole device information or for a particular resource. This resource
2548 should be only allowed when device meets a certain requirement, as it can create potential security
2549 issue.

2550 The delete is done using the device ID "id" as the tag in DELETE request query when all the
2551 resource information from the device is to be deleted. In the case of a specific resource then the
2552 DELETE request shall include the instance "ins" tag along with the device ID in the query.

2553 Selective deletion of information for individual resources is not possible the case where the RD
2554 pull the resource information. The publishing device can request a delete but only for all the
2555 resource information that the RD has pulled from that device. In this case, the DELETE request
2556 has the device ID "id" tag in the query.

2557 **11.3.6.3.4 Transfer resource information from one RD to another**

2558 When a publishing device identifies an RD that is better suited, it may decide to publish to that RD.
2559 Since the device should publish to only one RD at a time, the client shall ensure that previously
2560 published information is deleted from the currently used RD before publishing to the newly selected
2561 RD. The deletion of the resource may be done either by allowing the TTL to expire or explicitly
2562 deleting the resource information.

2563 RDs shall not communicate resource information between themselves. It is the client's
2564 responsibility to choose the RD and to manage the published resources.

2565 **11.3.6.4 Resource discovery**

2566 **11.3.6.4.1 /oic/res and retrieving of the resources**

2567 The /oic/res based discovery process remains the same as that in the absence of an RD.
2568 Resources may be discovered by retrieving the /oic/res resource by sending a multicast or unicast
2569 request. In the case of a multicast discovery request, an RD will respond for the device that hosts
2570 the resources. Clients shall be prepared to process duplicate resource information from more than
2571 one RD responding with the same information or from an RD and the hosting device (publishing
2572 the resource information) both responding to the request. Interaction with resources discovered
2573 using the RD is done using the same mechanism and methods as with resources discovered by
2574 retrieving the /oic/res resource of the device hosting the resources (e.g., connect to the resource
2575 and perform CRUDN operations on the resource).

2576 Resource Directory provides different /oic/res response according to requesting Clients, which
2577 indicate their preference with content format. OCF 1.0 Clients request with the "Content Format of
2578 "application/vnd.ocf+cbor", whereas the absence of the Content-Format indicates OIC 1.1 Clients.

2579 For OIC 1.1 Clients, /oic/res response includes to OIC 1.1 Link and anchor parameter has transfer
2580 protocol URI (e.g. coap URI), if present. The Resources hosted by the same Device are grouped
2581 together within a single JSON Object with "di" indicating the hosting Device. The Resources
2582 belonging to the responding RD may omit "anchor" parameter. However the Resources of other
2583 Devices shall include "anchor" parameter when "rel" value is "hosts".

2584 For example, a Resource Directory might return the following to OIC 1.1 clients:

```
2585 [
2586   {
2587     "di": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2588     "links": [
2589       {
2590         "href": "coap://[fe80::b1d4]:44444/oic/res",
2591         "rel": "self",
2592         "rt": ["oic.wk.res"],
2593         "if": ["oic.if.ll", "oic.if.baseline"],
2594         "p": {"bm": 3, "sec": true, "port": 11111}
2595       },
2596       {
2597         "href": "/oic/p",
2598         "rt": ["oic.wk.p"],
2599         "if": ["oic.if.r", "oic.if.baseline"],
2600         "p": {"bm": 3, "sec": true, "port": 11111}
2601       },
2602     ]
2603   }
2604 ]
```

```

2603     "href": "/oic/d",
2604     "rt": ["oic.wk.d"],
2605     "if": ["oic.if.r", "oic.if.baseline"],
2606     "p": {"bm": 3, "sec": true, "port": 11111}
2607   },
2608   {
2609     "href": "/oic/rd",
2610     "rt": ["oic.wk.rdpub"],
2611     "if": ["oic.if.baseline"],
2612     "p": {"bm": 3, "sec": true, "port": 11111}
2613   }
2614 ]
2615 },
2616 {
2617   "di": "88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
2618   "links": [
2619     {
2620       "anchor": "coaps://[fe80::b1d6]:22222",
2621       "href": "/oic/p",
2622       "rt": ["oic.wk.p"],
2623       "if": ["oic.if.r", "oic.if.baseline"],
2624       "p": {"bm": 3, "sec": true, "port": 11111}
2625     },
2626     {
2627       "anchor": "coaps://[fe80::b1d6]:22222",
2628       "href": "/oic/d",
2629       "rt": ["oic.wk.d", "oic.d.fan"],
2630       "if": ["oic.if.r", "oic.if.baseline"],
2631       "p": {"bm": 3, "sec": true, "port": 11111}
2632     },
2633     {
2634       "anchor": "coaps://[fe80::b1d6]:22222",
2635       "href": "/myFan",
2636       "rt": ["oic.r.switch.binary"],
2637       "if": ["oic.if.a", "oic.if.baseline"],
2638       "p": {"bm": 3, "sec": true, "port": 11111}
2639     }
2640   ]
2641 }
2642 ]

```

2643 For OCF 1.0 Clients, /oic/res response includes OCF 1.0 Link and anchor parameter has OCF URI.
2644 /oic/res response has the single array of OCF 1.0 Links to conform to IETF RFC 6690. Each Link
2645 shall carry "anchor" of the value OCF URI where the authority component of <deviceId> indicates
2646 the Device hosting the target Resource.

2647 The Resource Directory might return the following to clients that request with the Content Format
2648 of "application/vnd.ocf+cbor":

```

2649 [
2650   {
2651     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9/oic/res",
2652     "href": "/oic/res",
2653     "rel": "self",
2654     "rt": "oic.wk.res",
2655     "if": ["oic.if.ll", "oic.if.baseline"],
2656     "p": {"bm": 3},
2657     "eps": [{"ep": "coap://[fe80::b1d4]:44444"},
2658             {"ep": "coaps://[fe80::b1d4]:11111"}]
2659   },
2660   {
2661     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",

```

```

2662     "href": "/oic/p",
2663     "rt": ["oic.wk.p"],
2664     "if": ["oic.if.r", "oic.if.baseline"],
2665     "p": {"bm": 3},
2666     "eps": [{"ep": "coaps://[fe80::b1d4]:1111"}]
2667   },
2668   {
2669     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2670     "href": "/oic/d",
2671     "rt": ["oic.wk.d", "oic.d.light"],
2672     "if": ["oic.if.r", "oic.if.baseline"],
2673     "p": {"bm": 3},
2674     "eps": [{"ep": "coaps://[fe80::b1d4]:1111"}]
2675   },
2676   {
2677     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2678     "href": "/oic/rd",
2679     "rt": ["oic.wk.rdpub"],
2680     "if": ["oic.if.baseline"],
2681     "p": {"bm": 3},
2682     "eps": [{"ep": "coaps://[fe80::b1d4]:1111"}]
2683   },
2684   {
2685     "href": "/oic/p",
2686     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
2687     "rt": ["oic.wk.p"],
2688     "if": ["oic.if.r", "oic.if.baseline"],
2689     "p": {"bm": 3},
2690     "eps": [{"ep": "coaps://[fe80::b1d6]:2222"}]
2691   },
2692   {
2693     "href": "/oic/d",
2694     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
2695     "rt": ["oic.wk.d", "oic.d.fan"],
2696     "if": ["oic.if.r", "oic.if.baseline"],
2697     "p": {"bm": 3},
2698     "eps": [{"ep": "coaps://[fe80::b1d6]:2222"}]
2699   },
2700   {
2701     "href": "/myFan",
2702     "anchor": "ocf://88b7c7f0-4b51-4e0a-9faa-cfb439fd7f49",
2703     "rt": ["oic.r.switch.binary"],
2704     "if": ["oic.if.a", "oic.if.baseline"],
2705     "p": {"bm": 3},
2706     "eps": [{"ep": "coaps://[fe80::b1d6]:2222"}]
2707   }
2708 ]

```

2709

2710 11.4 Notification

2711 11.4.1 Overview

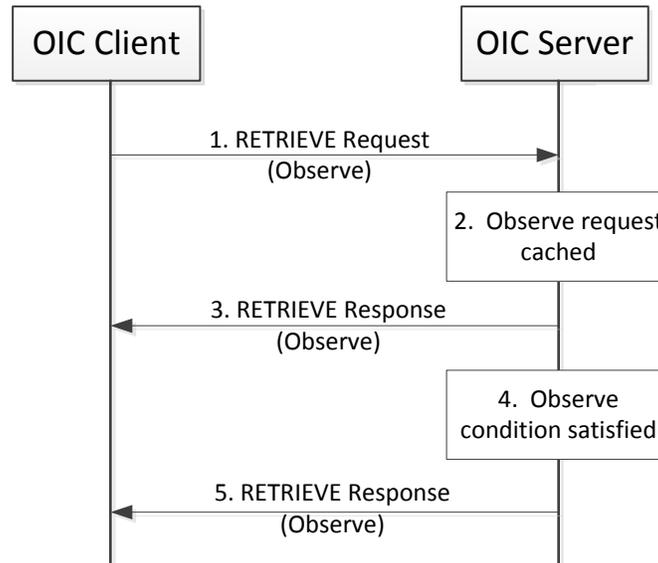
2712 A Server shall support NOTIFY operation to enable a Client to request and be notified of desired
 2713 states of one or more Resources in an asynchronous manner. Section 11.4.2 specifies the observe
 2714 mechanism in which updates are delivered to the requester.

2715 11.4.2 Observe

2716 In observe mechanism the Client utilizes the RETRIEVE operation to require the Server for updates
 2717 in case of Resource state changes. The Observe mechanism consists of five steps which are
 2718 depicted in Figure 29 and described below.

2719
2720

Note: the observe mechanism can only be used for a resource with a property of observable (section 7.3.2.2).



2721

2722

Figure 29. Observe Mechanism

2723 **11.4.2.1 RETRIEVE request with observe indication**

2724 The Client transmits a RETRIEVE request message to the Server to request updates for the
2725 Resource on the Server if there is a state change. The RETRIEVE request message carries the
2726 following parameters:

- 2727 • *fr*: Unique identifier of the Client
- 2728 • *to*: Resource that the Client is requesting to observe
- 2729 • *ri*: Identifier of the RETRIEVE request
- 2730 • *op*: RETRIEVE
- 2731 • *obs*: Indication for observe request

2732 **11.4.2.2 Processing by the Server**

2733 Following the receipt of the RETRIEVE request, the Server may validate if the Client has the
2734 appropriate rights for the requested operation and the properties are readable and observable. If
2735 the validation is successful, the Server caches the information related to the observe request. The
2736 Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial
2737 response and future responses in case of a change of state.

2738 **11.4.2.3 RETRIEVE response with observe indication**

2739 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
2740 message from a Client. The RETRIEVE response message shall include the following parameters.
2741 If validation succeeded, the response includes an observe indication. If not, the observe indication
2742 is omitted from the response which signals to the requesting client that registration for notification
2743 was not allowed.

2744 The RETRIEVE response message shall include the following parameters:

- 2745 • *fr*: Unique identifier of the Server
- 2746 • *to*: Unique identifier of the Client

- 2747 • *ri*: Identifier included in the RETRIEVE request
- 2748 • *cn*: Information resource representation as requested by the Client
- 2749 • *rs*: The result of the RETRIEVE operation
- 2750 • *obs*: Indication that the response is made to an observe request

2751 **11.4.2.4 Resource monitoring by the Server**

2752 The Server shall monitor the state the Resource identified in the observe request from the Client.
 2753 Anytime there is a change in the state of the observed resource, the Server sends another
 2754 RETRIEVE response with the observe indication. The mechanism does not allow the client to
 2755 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

2756 **11.4.2.5 Additional RETRIEVE responses with observe indication**

2757 The Server shall transmit updated RETRIEVE response messages following observed changes in
 2758 the state of the Resources indicated by the Client. The RETRIEVE response message shall include
 2759 the parameters listed in section 11.4.2.3.

2760 **11.4.2.6 Cancelling Observe**

2761 The Client can explicitly cancel observe by sending a RETRIEVE request without the observe
 2762 indication field to the same resource on Server which it was observing. For certain protocol
 2763 mappings, the client may also be able to cancel an observe by ceasing to respond to the
 2764 RETRIEVE responses.

2765 **11.5 Device management**

2766 **11.5.1 Overview**

2767 The Device Management includes the following functions:

- 2768 • Diagnostics and maintenance

2769 The device management functionalities specified in this version of specification are intended to
 2770 address the basic device management features. Addition of new device management features in
 2771 the future versions of the specification is expected.

2772 **11.5.2 Diagnostics and maintenance**

2773 The Diagnostics and Maintenance function is intended for use by administrators to resolve issues
 2774 encountered with the Devices while operating in the field. If diagnostics and maintenance is
 2775 supported by a Device, the Core Resource “/oic/mnt” shall be supported as described in Table 23.

2776 **Table 23. Optional diagnostics and maintenance device management Core Resources**

Pre-defined URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
/oic/mnt	Maintenance	oic.wk.mnt	oic.if.rw	The resource through which the device is maintained and can be used for diagnostic purposes. The resource properties exposed by “/oic/mnt” are listed in Table 24.	Device Management

2777

2778 Table 24 defines the oic.wk.mnt Resource Type. At least one of the Factory_Reset, and Reboot
 2779 properties shall be implemented.

Table 24. oic.wk.mnt Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Factory_Reset	fr	boolean			R, W	no	When writing to this Property: 0 – No action (Default*) 1 – Start Factory Reset After factory reset, this value shall be changed back to the default value (i.e., 0). After factory reset all configuration and state data will be lost. When reading this Property, a value of “1” indicates a pending factory reset, otherwise the value shall be “0” after the factory reset.
Reboot	rb	boolean			R, W	no	When writing to this Property: 0 – No action (Default) 1 – Start Reboot After Reboot, this value shall be changed back to the default value (i.e., 0)

2781

2782 Note: * - Default indicates the value of this property as soon as the device is rebooted or factory reset

2783

2784 **11.6 Scenes**2785 **11.6.1 Introduction**

2786 Scenes are a mechanism for automating certain operations.

2787 A scene is a static entity that stores a set of defined resource property values for a collection of
 2788 resources. Scenes provide a mechanism to store a setting over multiple Resources that may be
 2789 hosted by multiple separate Servers. Scenes, once set up, can be used by multiple Clients to recall
 2790 a setup.

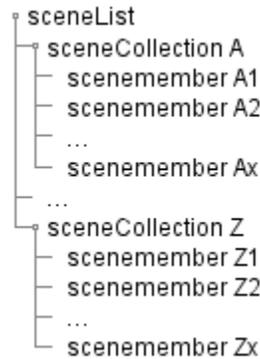
2791 Scenes can be grouped and reused, a group of scences is also a scene.

2792 In short, scenes are bundled user settings.

2793 **11.6.2 Scenes**2794 **11.6.2.1 Introduction**

2795 Scenes are described by means of resources. The scene resources are hosted by a Server and
 2796 the top level resource is listed in /oic/res. This means that a Client can determine if the scene
 2797 functionality is hosted on a Server via a RETRIEVE on /oic/res or via Resource discovery. The
 2798 setup of scenes is driven by Client interactions. This includes creating new scenes, and mappings
 2799 of Server resource properties that are part of a scene.

2800 The scene functionality is created by multiple resources and has the structure depicted in Figure
2801 30. The sceneList and sceneCollection resources are overloaded collection resources. The
2802 sceneCollection contains a list of scenes. This list contains zero or more scenes. The
2803 sceneMember resource contains the mapping between a scene and what needs to happen
2804 according to that scene on an indicated resource.



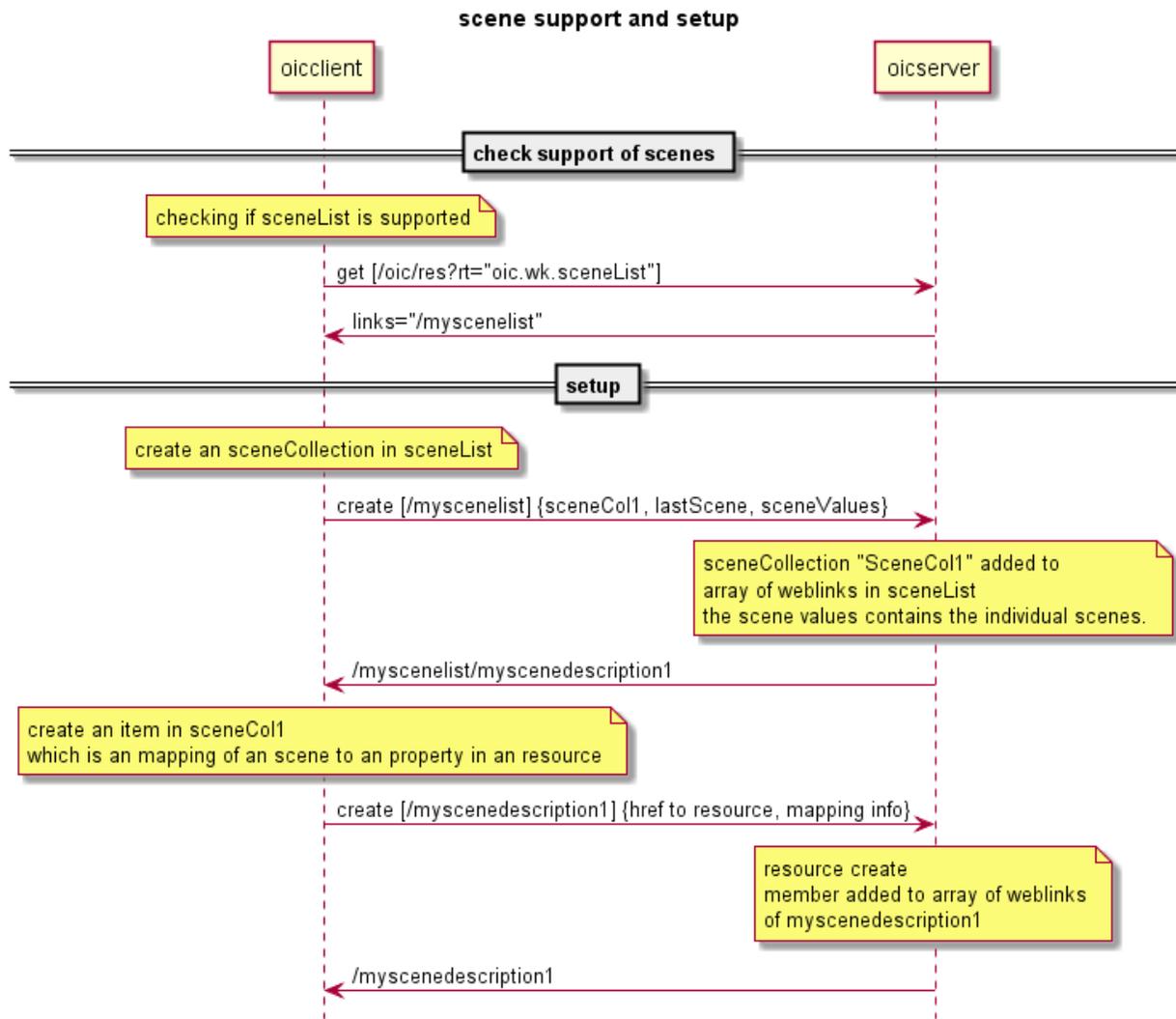
2805

2806

Figure 30 Generic scene resource structure

2807 **11.6.2.2 Scene creation**

2808 A Client desiring to interact with scenes needs to first determine if the server supports the scene
2809 feature; the sceneMembers of a scene do not have to be co-located on the server supporting the
2810 scene feature. This can be done by checking if /oic/res contains the rt of the sceneList resource.
2811 This is depicted in first steps of Figure 31. The sceneCollection is created by the Server using
2812 some out of bound mechanism, Client creation of scenes is not supported at this time. This will
2813 entail defining the scene with an applicable list of scene values and the mappings for each
2814 Resource being part of the scene. The mapping for each resource being part of the sceneCollection
2815 is described by a resource called sceneMember. The sceneMember resource contains the link to
2816 a resource and the mapping between the scene listed in the sceneValues property and the actual
2817 resource property value of the Resource indicated by the link.



2818

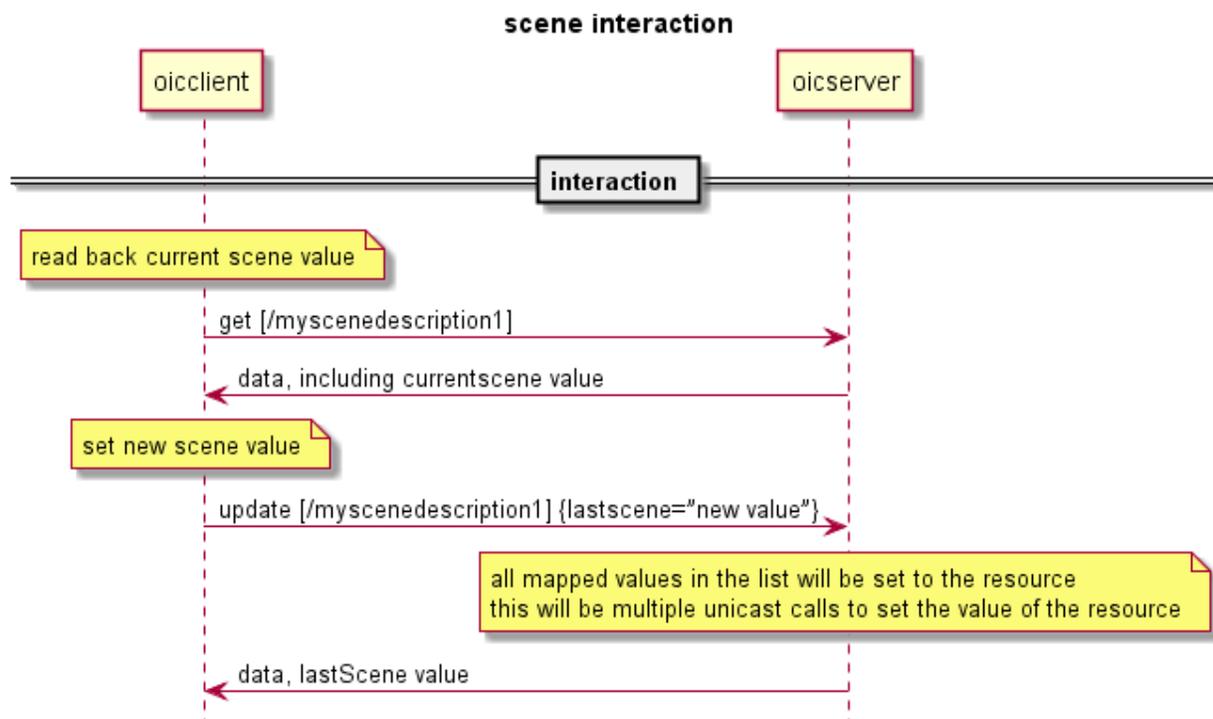
2819

Figure 31 Interactions to check Scene support and setup of specific scenes

2820

11.6.2.3 Interacting with Scenes

2821 All capable Clients can interact with scenes. The allowed scene values and the last applied scene
 2822 value can be retrieved from the server hosting the scene. The scene value shall be changed by
 2823 issuing an UPDATE operation with a payload that sets the lastScene property to one of the listed
 2824 allowed scene values. These steps are depicted in Figure 32. Note that the lastScene value does
 2825 not imply that the current state of all resources that are part of the scene will be at the mapped
 2826 value. This is due to that the setting the scene values are not modelled as actual states of the
 2827 system. This means that another Client can change just one resource being part of the scene
 2828 without having feedback that the state of the scene is changed.



2829

2830

Figure 32 Client interactions on a specific scene

2831 As described previously, a scene can reference one or more resources that are present on one or
 2832 more Servers. The scene members are re-evaluated each time a scene change takes place. This
 2833 evaluation is triggered by a Client that is either embedded as part of the Server hosting the scene,
 2834 or separate to the server having knowledge of the scene via a RETRIEVE operation, observing the
 2835 referenced resources using the mechanism described in section 11.4.2. During the evaluation the
 2836 mappings for the new scene value will be applied to the Server. This behaviour is depicted in
 2837 Figure 33.

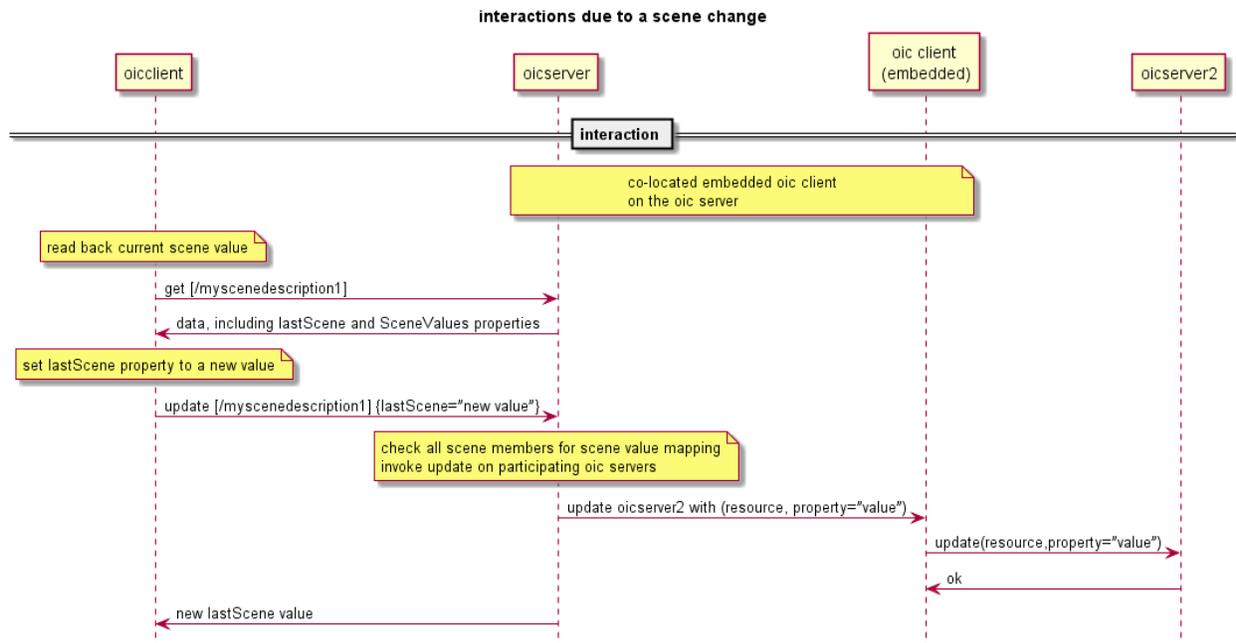


Figure 33 Interaction overview due to a Scene change

11.6.2.4 Summary of Resource Types defined for Scene functionality

Table 25 summarizes the list of Resource Types that are part of Scenes.

Table 25 list of Resource Types for Scenes

Friendly Name (informative)	Resource Type (rt)	Short Description	Section
sceneList	oic.wk.sceneList	Top Level collection containing sceneCollections	
sceneCollection	oic.wk.sceneCollection	Description of zero or more scenes	
sceneMember	oic.wk.sceneMember	Description of mappings for each specific resource part of the sceneCollection	

11.6.3 Security considerations

Creation of Scenes on a Server that is capable of this functionality is dependent on the ACLs applied to the resources and the Client having the appropriate permissions. Interaction between a Client (embedded or separate) and a Server that hosts the resource that is referenced as a scene member is contingent on the Client having appropriate permissions to access the resource on the host Server.

See OCF Security for details on the use of ACLs and also the mechanisms around Device Authentication that are necessary to ensure that the correct permissions exist for the Client to access the scene member resource(s) on the Server.

11.7 Icons

11.7.1 Overview

Icons are a primitive that are needed by various OCF subsystems, such as bridging. An optional Resource Type of “oic.r.icon” has been defined to provide a common representation of an icon Resource that can be used by Devices.

2857 **11.7.2 Resource**

2858 The icon Resource is as defined in Table 26.

2859 **Table 26. Optional Icon Core Resource**

URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/example/oic/icon	Icon	oic.r.icon	oic.if.r	The Resource through which the Device can obtain icon images. The Resource properties exposed by "/example/oic/mnt" are listed in Table 27.	Icon

2860

2861 Table 27 defines the details for the "oic.r.icon" Resource Type.

2862 **Table 27. oic.r.icon Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Mime Type	mimetype	string			R	yes	Specifies the format (media type) of the icon. It should be a template string as specified in IANA Media Types Assignment
Width	width	integer	>= 1		R	yes	Width of the icon in pixels greater than or equal to 1.
Height	height	integer	>= 1		R	yes	Height of the icon in pixels greater than or equal to 1.
Icon	media	uri			R	yes	URI to the location of the icon image.

2863

2864 **11.8 Introspection**

2865 **11.8.1 Overview**

2866 Introspection is a mechanism to announce the capabilities of Resources hosted on the Device.

2867 The intended usage of the Introspection Device Data is to enable dynamic clients. E.g. clients that
 2868 can use the Introspection Device Data to generate dynamically an UI or dynamically create
 2869 translations of the hosted Resources to another eco-system. Other usages of the Introspection is
 2870 that the information can be used to generate client code. The Introspection Device Data is designed
 2871 to augment the existing data already on the wire. This means that existing mechanism needs to
 2872 be used to get a full overview of what is implemented in the Device. For example the Introspection
 2873 Device Data does not convey information about observe, since that is already conveyed with the
 2874 "p" property on the links in "/oic/res" (see section 7.8.2.1.2).

2875 The Introspection Device Data is recommended to be conveyed as "static" data. Meaning that the
 2876 data does not change during the uptime of a Device. However when the data is not static the
 2877 Introspection Resource shall indicate to be observable and the url property value of
 2878 oic.wk.introspection Resource shall change to indicate that the Introspection Device Data is
 2879 changed.

2880 The Introspection Device Data describes the Resources that make up the Device. For the complete
 2881 list of included Resources Table 13. The Introspection Device Data is described as a swagger2.0

2882 in JSON format file. The swagger2.0 file will contain the description of the Resources as defined
2883 below: All Resources with the next remarks:

- 2884 • The URLs of the Resources in the Introspection Device Data shall be without the endpoint
2885 description, e.g. it shall not be a full URL but only the relative path from the endpoint. The
2886 relative path shall be the same as being conveyed by “/oic/res”.
- 2887 • “/oic/res” Resource shall not be listed in the Introspection Device Data.
- 2888 • The Resources “/oic/d”, “/oic/p” and the security Resources are allowed to be present in the
2889 Introspection Device Data, but are not required. The “/oic/d”, “/oic/p”, “/oic/res” and the security
2890 Resources shall be included when vendor defined or optional properties are implemented.
- 2891 • All other Resources are required to be listed in the Introspection Device Data.
- 2892 • Per Resource it will include:
 - 2893 ○ All Implemented Methods
 - 2894 ○ Per Supported Method:
 - 2895 ■ Implemented queryParameters per Method.
 - 2896 • This includes the supported interfaces ("if") as enum value.
 - 2897 ■ Schemas of the payload for the request and response bodies of the Method
 - 2898 ■ The schema data shall be conveyed by the swagger schema object as
2899 defined in the parameters section.
 - 2900 ■ The swagger2.0 schema object shall comply with:
 - 2901 • The schemas shall be fully resolved, e.g. no references shall exist
2902 outside the swagger file.
 - 2903 • The schemas shall list which interfaces are supported on the method.
 - 2904 • The schemas shall list if a property is optional or required.
 - 2905 • The schemas shall indicate if a property is read only or read-write
 - 2906 ○ By means of the readOnly schema tag belonging to the
2907 property
 - 2908 • The default value of the “rt” property shall be used to indicate the
2909 supported Resource Types.
 - 2910 • oneOf and anyOf constructs are allowed to be used as part of an
2911 swagger2.0 schema object.

2912 Dynamic Resources (e.g. Resources that can be created up on a request by a Client) shall have
2913 an URL definition which contains a URL identifier (e.g. using the {} syntax). An URL with {} identifies
2914 that the Resource definition applies to the whole group of Resources that can be created. The
2915 actual path can contain the collection node that links to the Resource.

2916 Example of an URL with identifiers:

2917 /SceneListResURI/{SceneCollectionResURI}/{SceneMemberResURI};

2918 When different Resource Types are allowed to be created in a collection, then the different
2919 schemas for the create method shall define all possible Resource Types that can be created. The
2920 schema construct oneOf allows the definition of a schema with selectable Resources. The oneOf
2921 construct allows the integration of all schemas and that only one existing sub schemas shall be
2922 used to indicate the definition of the Resource that can be created.

2923 Example usage of oneOf JSON schema construct:

```

2924 {
2925   "oneOf": [
2926     { <<subschema 1 definition>> },
2927     { << sub schema 2 definition >> }
2928   ...
2929   ]
2930 }

```

2931

2932 A Client using the Introspection Device Data of a Device should check the version of the supported
 2933 Introspection Device Data of the Device. The swagger version is indicated in each file with the tag
 2934 "swagger". Example of the 2.0 supported version of the tag is: "swagger": "2.0". Later versions of
 2935 the spec may reference newer versions of the swagger specification, for example 3.0.

2936 A Server shall support one Resource with a Resource Type of "oic.wk.introspection" as defined in
 2937 Table 28. The Resource with a Resource Type of "oic.wk.introspection" shall be included in the
 2938 Resource "/oic/res".

2939 **Table 28. Introspection Resource**

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

2940

2941 Table 29 defines oic.wk.introspection Resource Type.

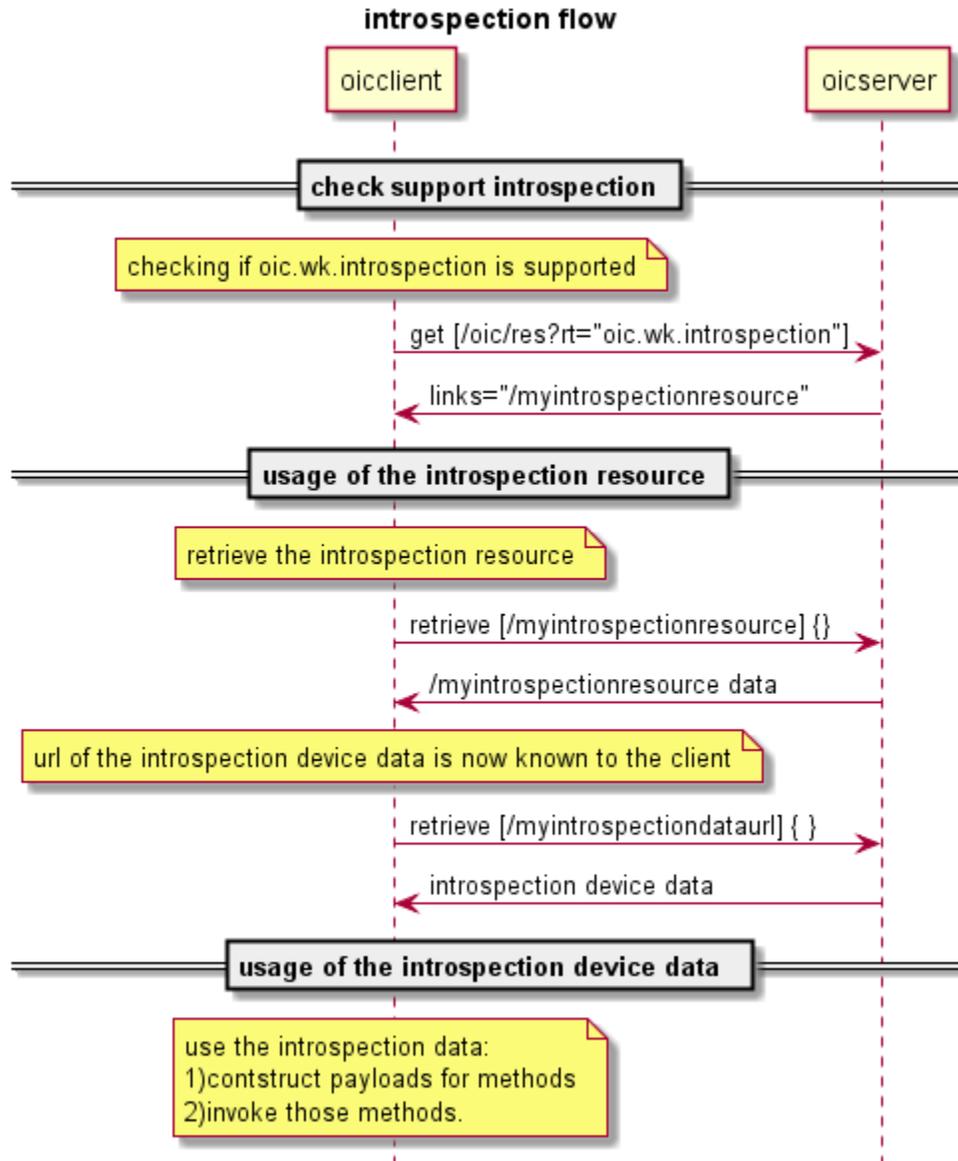
2942 **Table 29. oic.wk.introspection Resource Type definition**

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

2943 **11.8.2 Usage of introspection**

2944 The Introspection Device Data is retrieved in the following steps:

- 2945 1) Check if the Introspection Resource is supported and retrieve the URL of the Resource.
 2946 2) Retrieve the contents of the Introspection Resource
 2947 3) Download the Introspection Device Data from the URL specified the Introspection Resource.
 2948 4) Usage of the Introspection Device Data by the Client



2949 **Figure 34 Interactions to check Introspection support and download the Introspection**
 2950 **Device Data.**
 2951

2952 **12 Messaging**

2953 **12.1 Introduction**

2954 This section specifies the protocol messaging mapping to the CRUDN messaging operations
 2955 (Section 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols
 2956 is expected in later version of this specification. All the property information from the resource

2957 model shall be carried within the message payload. This payload shall be generated in the resource
 2958 model layer and shall be encapsulated in the data connectivity layer. The message header shall
 2959 only be used to describe the message payload (e.g., verb, mime-type, message payload format),
 2960 in addition to the mandatory header fields defined in messaging protocol (e.g., CoAP) specification.
 2961 If the message header does not support this, then this information shall also be carried in the
 2962 message payload. Resource model information shall not be included in the message header
 2963 structure unless the message header field is mandatory in the messaging protocol specification.

2964 **12.2 Mapping of CRUDN to CoAP**

2965 **12.2.1 Overview**

2966 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in section
 2967 12.2.3. A Device implementing CoAP shall conform to IETF RFC 7641 to implement the CoAP
 2968 Observe option. Support for CoAP block transfer when the payload is larger than the MTU is
 2969 defined in section 12.2.8.

2970 **12.2.2 URIs**

2971 An OCF: URI is mapped to a coap: URI by replacing the scheme name 'oic' with 'coap' if unsecure
 2972 or 'coaps' if secure before sending over the network by the requestor. Similarly on the receiver
 2973 side, the scheme name is replaced with 'oic'.

2974 **12.2.3 CoAP method with request and response**

2975 **12.2.3.1 Overview**

2976 Every request has a CoAP method that realizes the request. The primary methods and their
 2977 meanings are shown in Table 30, which provides the mapping of GET/PUT/POST/DELETE
 2978 methods to CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides
 2979 the generic behaviours when using these methods, however resource interfaces may modify these
 2980 generic semantics.
 2981

2982 **Table 30. CoAP request and response**

Method for CRUDN	(mandatory) Request data	(mandatory) Response data
GET for RETRIEVE	<ul style="list-style-type: none"> - Method code: GET (0.01) - Request URI: an existing URI for the Resource to be retrieved 	<ul style="list-style-type: none"> - Response code: success (2.xx) or error (4.xx or 5.xx) - Payload: Resource representation of the target Resource (when successful)
POST for CREATE	<ul style="list-style-type: none"> - 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 	<ul style="list-style-type: none"> - Response code: success (2.xx) or error (4.xx or 5.xx) - Payload: the URI of the newly created Resource (when successful).
PUT for CREATE	<ul style="list-style-type: none"> - Method code: PUT (0.03) - Request URI: a new URI for the Resource to be created. - Payload: Resource presentation of the Resource to be created. 	<ul style="list-style-type: none"> - Response code: success (2.xx) or error (4.xx or 5.xx)
POST for UPDATE	<ul style="list-style-type: none"> - Method code: POST (0.02) - Request URI: an existing URI for the Resource to be updated. - Payload: representation of the Resource to be updated. 	<ul style="list-style-type: none"> - Response Code: success (2.xx) or error (4.xx or 5.xx)

DELETE for DELETE	<ul style="list-style-type: none"> - Method code: DELETE (0.04) - Request URI: an existing URI for the Resource to be deleted. 	<ul style="list-style-type: none"> - Response code: success (2.xx) or error (4.xx or 5.xx)
--------------------------	--	--

2983

2984 **12.2.3.2 CREATE with POST or PUT**

2985 **12.2.3.2.1 With POST**

2986 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
 2987 existing Resource on the Server that is processing the request. If no such Resource is present,
 2988 the Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall
 2989 use an existing request URI which identifies the Resource on the Server responsible for creation.
 2990 The URI of the created Resource is determined by the Server and provided to the Client in the
 2991 response.

2992 A Client shall include the representation of the new Resource in the request payload. The new
 2993 resource representation in the payload shall have all the necessary properties to create a valid
 2994 Resource instance, i.e. the created Resource should be able to properly respond to the valid
 2995 Request with mandatory Interface (e.g., GET with ?if=oic.if.baseline).

2996 Upon receiving the POST request, the Server shall either

- 2997 • create the new Resource with a new URI, respond with the new URI for the newly created
 2998 Resource and a success response code (2.xx); or
- 2999 • respond with an error response code (4.xx or 5.xx).

3000 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or
 3001 guaranteed.

3002 **12.2.3.2.2 With PUT**

3003 PUT shall be used to create a new Resource or completely replace the entire representation of an
 3004 existing Resource. The resource representation in the payload of the PUT request shall be the
 3005 complete representation. PUT for CREATE shall use a new request URI identifying the new
 3006 Resource to be created.

3007 The new resource representation in the payload shall have all the necessary properties to create
 3008 a valid Resource instance, i.e. the created Resource should be able to properly respond to the
 3009 valid Request with mandatory Interface (e.g. GET with ?if=oic.if.baseline).

3010 Upon receiving the PUT request, the Server shall either

- 3011 • create the new Resource with the request URI provided in the PUT request and send back a
 3012 response with a success response code (2.xx); or
- 3013 • respond with an error response code (4.xx or 5.xx).

3014 PUT is an unsafe method but it is idempotent, thus when a PUT request is repeated the outcome
 3015 is the same each time.

3016 **12.2.3.3 RETRIEVE with GET**

3017 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of
 3018 the target Resource identified by the request URI.

3019 Upon receiving the GET request, the Server shall either

- 3020 • send back the response with the representation of the target Resource with a success response
 3021 code (2.xx); or

3022 • respond with an error response code (4.xx or 5.xx) or ignore it (e.g. non-applicable multicast
3023 GET).

3024 GET is a safe method and is idempotent.

3025 12.2.3.4 UPDATE with POST

3026 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
3027 existing Resource on the Server that is processing the request. If no such Resource is present,
3028 the Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE
3029 Property values of an existing Resource (see Sections 3.1.32 and 8.4.2).

3030 Upon receiving the request, the Server shall either

- 3031 • apply the request to the Resource identified by the request URI in accordance with the applied
3032 interface (i.e. POST for non-existent Properties is ignored) and send back a response with a
3033 success response code (2.xx); or
- 3034 • respond with an error response code (4.xx or 5.xx). Note that if the representation in the
3035 payload is incompatible with the target Resource for POST using the applied interface (i.e. the
3036 "overwrite" semantic cannot be honored because of read-only property in the payload), then
3037 the error response code 4.xx shall be returned.

3038 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or
3039 guaranteed.

3040 12.2.3.5 DELETE with DELETE

3041 DELETE shall be used for DELETE operation. The DELETE method requests that the resource
3042 identified by the request URI be deleted.

3043 Upon receiving the DELETE request, the Server shall either

- 3044 • delete the target Resource and send back a response with a success response code (2.xx); or
- 3045 • respond with an error response code (4.xx or 5.xx).

3046 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

3047
3048

3049 12.2.4 Content-Format negotiation

3050 The OCF Framework mandates support of CBOR, however it allows for negotiation of the payload
3051 body if more than one Content-Format (e.g. CBOR and JSON) is supported by an implementation.
3052 In this case the Accept Option defined in section 5.10.4 of IETF RFC 7252 shall be used to indicate
3053 which Content-Format (e.g. JSON) is requested by the Client.

3054 The Content-Formats supported are shown in Table 31.

3055

Table 31. OCF Content-Formats

Media Type	ID
application/cbor	60
application/vnd.ocf+cbor	10000

3056 Clients shall include a Content-Format Option in every message that contains a payload. Servers
3057 shall include a Content-Format Option for all success (2.xx) responses with a payload body. Per
3058 IETF RFC 7252 section 5.5.1, Servers shall include a Content-Format Option for all error (4.xx or
3059 5.xx) responses with a payload body unless they include a Diagnostic Payload; error responses

3060 with a Diagnostic Payload do not include a Content-Format Option. The Content-Format Option
 3061 shall use the ID column numeric value from Table 31. An OCF vertical may mandate a specific
 3062 Content-Format Option.

3063 Clients shall also include an Accept Option in every request message. The Accept Option shall
 3064 indicate the required Content-Format as defined in Table 31 for response messages. The Server
 3065 shall return the required Content-Format if available. If the required Content-Format cannot be
 3066 returned, then the Server shall respond with an appropriate error message.

3067 **12.2.5 Content-Format Version information**

3068 Servers and Clients shall include the Content-Format Version in both request and response
 3069 messages with a payload. Clients shall include the Accept Version in request messages. The
 3070 Content-Format Version and Accept Version are specified as Option Numbers in the CoAP header
 3071 as shown in Table 32.

3072 **Table 32. Content-Format Version and Accept Version Option Numbers**

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

3073 The value of the Accept Version and the Content-Format Version is a two-byte unsigned integer
 3074 that is used to define the major, minor and sub versions. The major and minor versions are
 3075 represented by 5 bits and the sub version is represented by 6 bits as shown in Table 33.

3076 **Table 33. Accept Version and the Content-Format Version Representation**

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

3077 Table 34 illustrates several examples:

3078 **Table 34. Examples of Content-Format Version and Accept Version Representation**

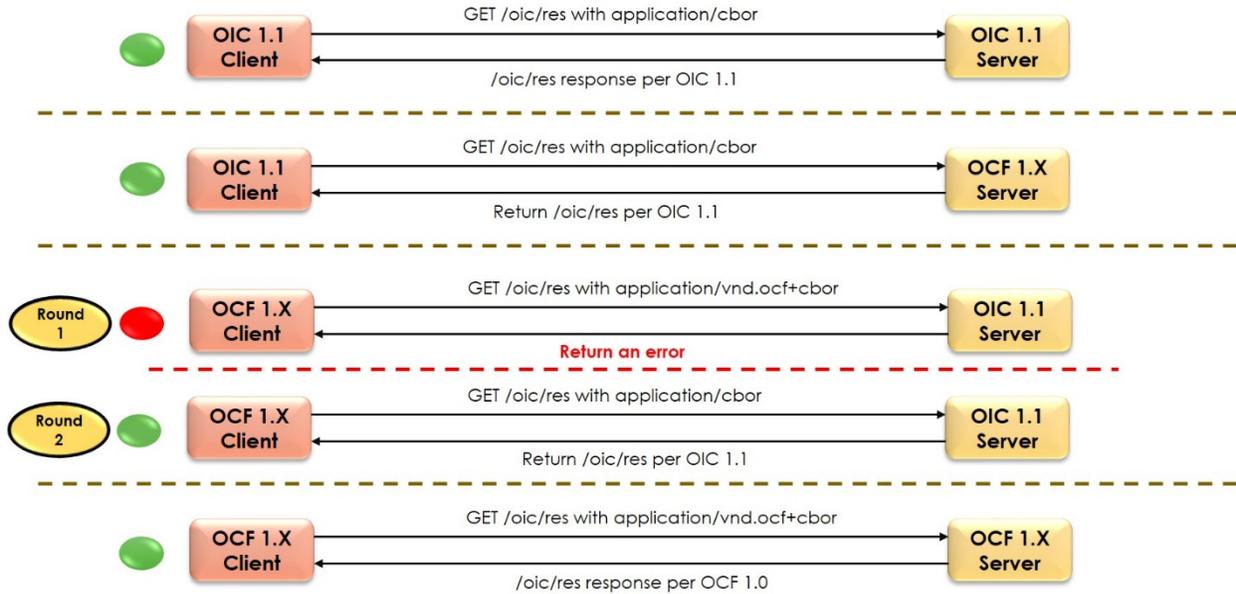
OCF version	Binary representation	Integer value
1.0.0	0000 1000 0000 0000	2048
1.1.0	0000 1000 0100 0000	2112

3079 The Accept Version and Content-Format Version for this version of the specification shall be 1.0.0
 3080 (i.e. 0b0000 1000 0000 0000).

3081 **12.2.6 Content-Format policy**

3082 To maintain compatibility between devices implemented to different versions of this specification,
 3083 Devices shall follow the policy as described in Figure 35.

3084



3085

3086

Figure 35 Content-Format Policy

3087 All Devices shall support the current and all previous Content-Format Option and Versions. Clients
 3088 shall send discovery request messages with the current and all previous Content-Format and
 3089 Versions until it discovers all Servers in the network.

3090 **12.2.7 CRUDN to CoAP response codes**

3091 The mapping of CRUDN operations response codes to CoAP response codes are identical to the
 3092 response codes defined in IETF RFC 7252.

3093 **12.2.8 CoAP block transfer**

3094 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT
 3095 devices. However scenarios can be envisioned in which an application needs to transfer larger
 3096 payloads.

3097 CoAP block-wise transfer as defined in <https://tools.ietf.org/html/rfc7721>

3098 IETF RFC 7959 shall be used by all Servers which generate a content payload that would exceed
 3099 the size of a CoAP datagram as the result of handling any defined CRUDN operation.

3100 Similarly, CoAP block-wise transfer as defined in <https://tools.ietf.org/html/rfc7721>

3101 IETF RFC 7959 shall be supported by all Clients. The use of block-wise transfer is applied to
 3102 both the reception of payloads as well as transmission of payloads that would exceed the size of
 3103 a CoAP datagram.

3104 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the
 3105 same reliability setting (i.e. all confirmable or all non-confirmable).

3106 A Client may support both the block1 (as descriptive) and block2 (as control) options as
 3107 described by IETF RFC 7959 A Server may support both the block1 (as control) and block2 (as
 3108 descriptive) options as described by <https://tools.ietf.org/html/rfc7721>

3109 IETF RFC 7959.

3110 **12.3 CoAP serialization over TCP**

3111 **12.3.1.1 Introduction**

3112 In environments where TCP is already available, CoAP can take advantage of it to provide
3113 reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For
3114 example, consider a cloud application acting as a Client and the Server is located at the user's
3115 home. The Server which already support CoAP as a messaging protocol (e.g., Smart Home vertical
3116 profile) could easily support CoAP serialization over TCP rather than adding another messaging
3117 protocol. A Device implementing CoAP Serialization over TCP should conform to IETF draft-ietf-
3118 core-coap-tcp-tls-07.

3119 **12.3.1.2 Indication of support**

3120 If UDP is blocked, clients depend on the pre-configured details on the device to find support for
3121 CoAP over TCP. If UDP is not-blocked, a Device which supports CoAP serialization over TCP shall
3122 populate the Messaging Protocol (mpro) property in /oic/res with the value "coap+tcp" or
3123 "coaps+tcp" to indicate that the device supports messaging protocol as specified by section 11.3.4.

3124 **12.3.1.3 Message type and header**

3125 The message type transported between Client and Server shall be a non-confirmable message
3126 (NON). The protocol stack used in this scenario should be as described in section 3 in IETF draft-
3127 ietf-core-coap-tcp-tls-07.

3128 The CoAP header as described in figure 6 in IETF draft-ietf-core-coap-tcp-tls-07 should be used
3129 for messages transmitted between a Client and a Server. A Device should use "Alternative L3" as
3130 defined in IETF draft-ietf-core-coap-tcp-tls-07.

3131 **12.3.1.4 URI scheme**

3132 The URI scheme used shall be as defined in section 6 in IETF draft-ietf-core-coap-tcp-tls-07].

3133 For the "coaps+tcp" URI scheme the "TLS Application Layer Protocol Negotiation Extension"
3134 IETF RFC 7301 shall be used.

3135 **12.3.1.5 KeepAlive**

3136 **12.3.1.5.1 Overview**

3137 In order to ensure that the connection between a Device is maintained, when using CoAP
3138 serialization over TCP, a Device that initiated the connection should send application layer
3139 KeepAlive messages. The reasons to support application layer KeepAlive are as follows:

- 3140 • TCP KeepAlive only guarantees that a connection is alive at the network layer, but not at the
3141 application layer
- 3142 • Interval of TCP KeepAlive is configurable only using kernel parameters, and is OS dependent
3143 (e.g., 2 hours by default in Linux)

3144 **12.3.1.5.2 KeepAlive Mechanism**

3145 Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. A Server shall
3146 support a resource of type oic.wk.ping as defined in Table 35.

3147 **Table 35. Ping resource**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/ping	Ping	oic.wk.ping	oic.if.rw	The resource using which a Client keeps its Connection with a Server active.	KeepAlive

				The resource properties exposed by /oic/ping are listed in Table 36.	
--	--	--	--	--	--

3148

3149 Table 36 defines oic.wk.ping Resource Type.

3150

Table 36. oic.wk.ping Resource Type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interval	in	integer	minutes		R,W	yes	The time interval for which connection shall be kept alive and not closed. Default value is 0.

3151 The following steps detail the KeepAlive mechanisms for a Client and Server:

3152 1) A Client which wants to keep the connection with a Server alive shall send a POST request to
3153 /oic/ping resource on the Server updating its connection Interval.

3154 a. This time interval shall start from 2 minutes and increases in multiples of 2
3155 up to a maximum of 64 minutes. It stays at 64 minutes from that point.

3156 5) A Server receiving this ping request shall respond within 1 minute.

3157 6) If a Client does not receive the response within 1 minute, it shall terminate the connection.

3158 7) If a Server does not receive a POST request to ping resource within the specified "interval"
3159 time, the Server shall terminate the connection.

3160 An example of the KeepAlive mechanism is as follows:

- 3161 • Client → Server: POST/oic/ping {interval: 2}
- 3162 • Server → Client: 2.03 valid

3163 12.4 Payload Encoding in CBOR

3164 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to
3165 JSON from CBOR in accordance with IETF RFC 7049 section 4 unless otherwise specified in this
3166 section.

3167 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types
3168 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-
3169 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation
3170 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer
3171 numbers shall be within the closed interval $[-2^{53}, 2^{53}]$. Properties defined as a JSON number
3172 should be encoded as integers whenever possible; if this is not possible Properties defined as a
3173 JSON number should use single-precision if the loss of precision does not affect the quality of
3174 service, otherwise the Property shall use double-precision.

3175

3176 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values
3177 in any position. If a property defined as a JSON integer is received encoded other than as an
3178 integer, the implementation may reject this encoding using a final response as appropriate for the
3179 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a property is
3180 defined as a JSON number an implementation shall accept integers, single- and double-precision
3181 floating point.

3182 13 Security

3183 The details for handling security and privacy are specified in [OCF Security].

DRAFT

3185
3186
3187
3188

Annex A (informative)

Operation Examples

3189 A.1 Introduction

3190 This section describes some example scenarios using sequence of operations between the entities
3191 involved. In all the examples below “Light” is a Server and “Smartphone” is a Client. In one of the
3192 scenario “Garage” additionally acts as a Server. All the examples are based on the following
3193 example resource definitions:

3194 `rt=oic.example.light` with Resource Type definition as illustration in Table 37.

3195 **Table 37. oic.example.light Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
on-off	of	boolean			R, W	yes	On/Off Control: 0 = Off 1 = On
dim	dm	integer	0-255		R, W	yes	Resource which can take a range of values minimum being 0 and maximum being 255

3196

3197 `rt=oic.example.garagedoor` with Resource Type definition as illustration in Table 38.

3198 **Table 38. oic.example.garagedoor Resource Type definition**

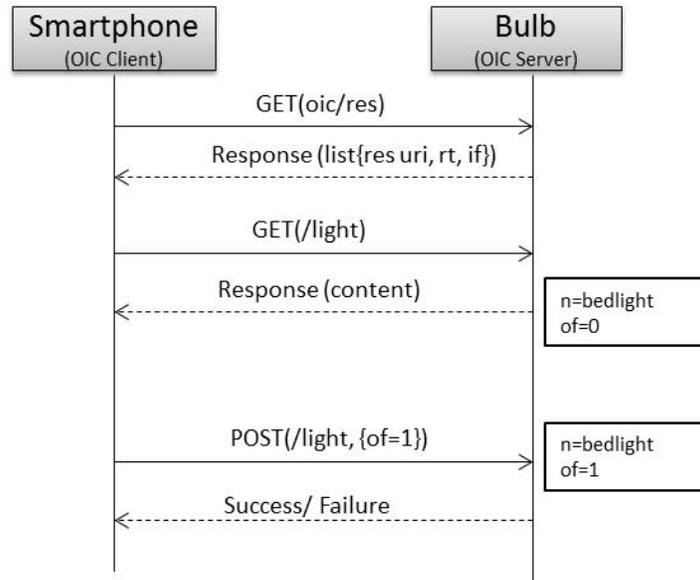
Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
open-close	oc	boolean			R, W	yes	Open/Close Control: 0 = Open 1 = Close

3199

3200 `/oic/mnt` (`rt=oic.wk.mnt`) used in below examples is defined in section 11.5.2.

3201 A.2 When at home: From smartphone turn on a single light

3202 This sequence highlights (Figure 36) the discovery and control of an OCF light resource from an
3203 OCF smartphone.



3204

3205

Figure 36. When at home: from smartphone turn on a single light

3206 Discovery request can be sent to “All OCF Nodes” Multicast address FF0X::158 or can be sent
3207 directly to the IP address of device hosting the light resource.

3208 1) Smartphone sends a GET request to /oic/res resource to discover all resources hosted on
3209 targeted end point

3210 8) The end point (bulb) responds with the list of Resource URI, Resource Type and
3211 Interfaces supported on the end point (one of the resource is '/light' whose
3212 rt=oic.example.light)

3213 9) Smartphone sends a GET request to '/light' resource to know its current state

3214 10) The end point responds with representation of light resource ({n=bedlight;of=0})

3215 11) Smartphone changes the 'of' property of the light resource by sending a POST
3216 request to '/light' resource ({of=1})

3217 12) On Successful execution of the request, the end point responds with the changed
3218 resource representation. Else, error code is returned. Details of the error codes are defined
3219 in section 12.2.7.

3220 **A.3 GroupAction execution**

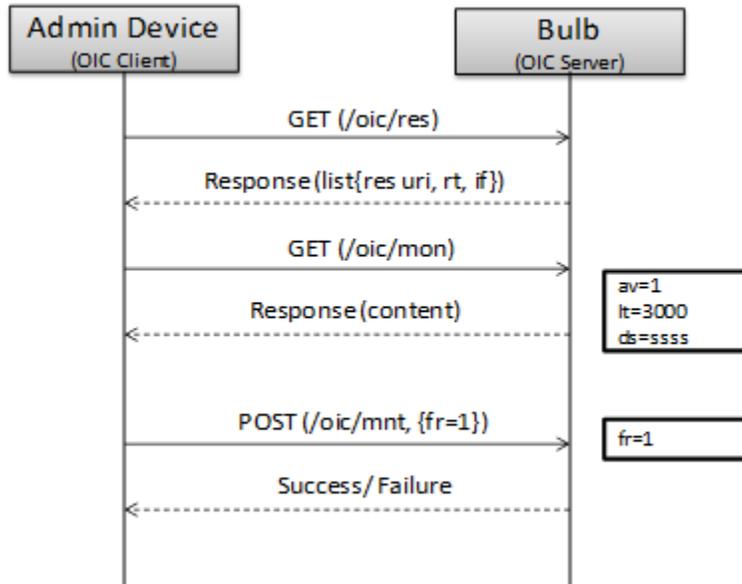
3221 This example will be added when groups feature is added in later version of specification

3222 **A.4 When garage door opens, turn on lights in hall; also notify smartphone**

3223 This example will be added when scripts feature is added in later version of specification

3224 **A.5 Device management**

3225 This sequence highlights (Figure 37) the device management function of maintenance.



3227

3228

Figure 37. Device management (maintenance)

3229 **Pre-Condition:** Admin device has different security permissions and hence can perform device
3230 management operations on the Device

3231 1) Admin device sends a GET request to /oic/res resource to discover all resources hosted on a
3232 targeted end point (in this case Bulb)

3233 13) The end point (bulb) responds with the list of Resource URI, Resource Type and Interfaces
3234 supported on the end point (one of the resources is /oic/mnt whose rt=oic.wk.mnt)

3235 14) Admin Device changes the 'fr' property of the maintenance resource by
3236 sending a POST request to /oic/mnt resource ({fr=1}). This triggers a factory reset of the
3237 end point (bulb)

3238 15) On successful execution of the request, the end point responds with the changed
3239 resource representation. Else, error code is returned. Details of the error codes are defined
3240 in section 12.2.7.

3241
3242
3243
3244

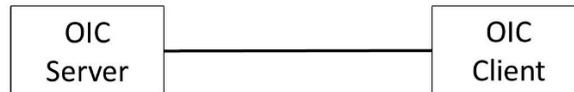
Annex B (informative)

OCF interaction scenarios and deployment models

3245 B.1 OCF interaction scenarios

3246 A Client connects to one or multiple Servers in order to access the resources provided by those
3247 Servers. The following are scenarios representing possible interactions among Roles:

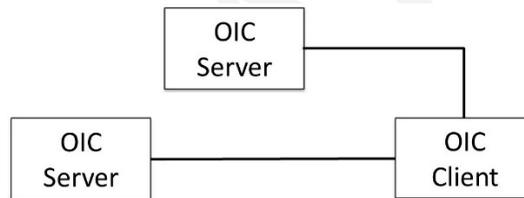
- 3248 • Direct interaction between Client and Server (Figure 38). In this scenario the Client and the
3249 Server directly communicate without involvement of any other Device. A smartphone which
3250 controls an actuator directly uses this scenario.



3251

3252 **Figure 38. Direct interaction between Server and Client**

- 3253 • Interaction between Client and Server using another server (Figure 39). In this scenario,
3254 another Server provides the support needed for the Client to directly access the desired
3255 resource on a specific Server. This scenario is used for example, when a smartphone first
3256 accesses a discovery server to find the addressing information of a specific appliance, and
3257 then directly accesses the appliance to control it.



3258

3259 **Figure 39. Interaction between Client and Server using another Server**

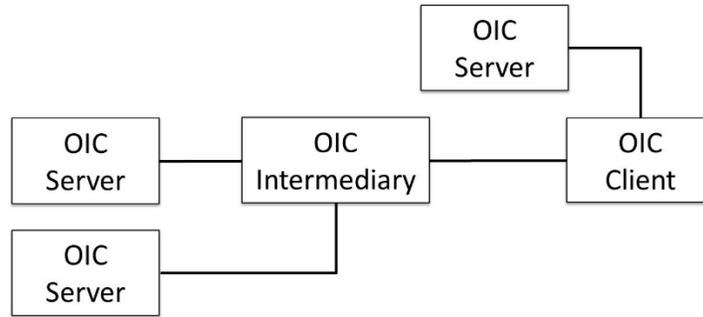
- 3260 • Interaction between Client and Server using Intermediary (Figure 40). In this scenario an
3261 Intermediary facilitates the interaction between the Client and the Server. A smartphone which
3262 controls appliances in a smart home via MQTT broker uses this scenario.



3263

3264 **Figure 40. Interaction between Client and Server using Intermediary**

- 3265 • Interaction between Client and Server using support from multiple Servers and intermediary
3266 (Figure 41). In this scenario, both Server and Intermediary roles are present to facilitate the
3267 transaction between the Client and a specific Server. An example scenario is when a
3268 smartphone first accesses a Resource Directory (RD) server to find the address to a specific
3269 appliance, then utilizes MQTT broker to deliver a command message to the appliance. The
3270 smartphone can utilize the mechanisms defined in CoRE Resource Directory such as default
3271 location, anycast address or DHCP to discover the Resource Directory information.

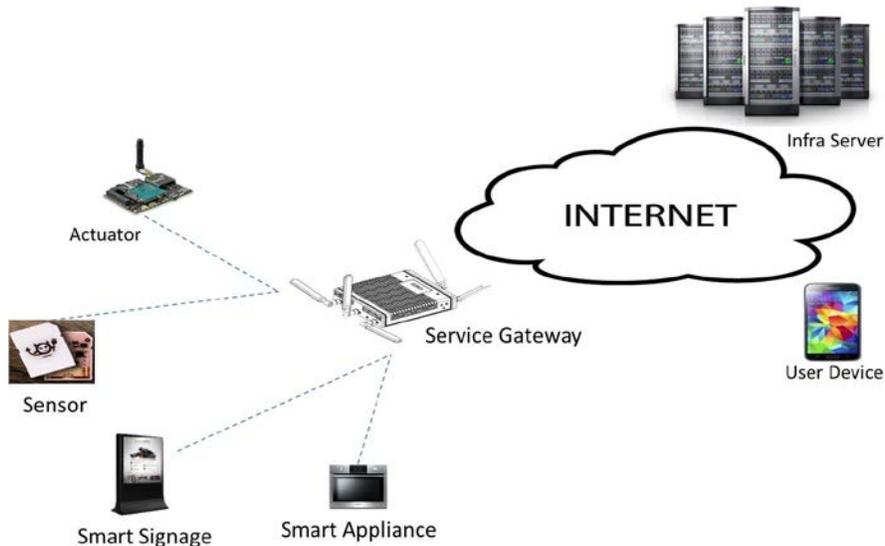


3272

3273 **Figure 41. Interaction between Client and Server using support from multiple Servers and**
 3274 **Intermediary**

3275 **B.2 Deployment model**

3276 In deployment, Devices are deployed and interact via either wired or wireless connections. Devices
 3277 are the physical entities that may host resources and play one or more Roles. There is no constraint
 3278 on the structure of a deployment or number of Devices in it. Architecture is flexible and scalable
 3279 and capable of addressing large number of devices with different device capabilities, including
 3280 constrained devices which have limited memory and capabilities. Constrained devices are defined
 3281 and categorized in [TCNN].



3282

3283 **Figure 42. Example of Devices**

3284 Figure 42 depicts a typical deployment and set of Devices, which may be divided in the following
 3285 categories:

- 3286 • **Things:** Networked devices which are able to interface with physical environments. Things are
 3287 the devices which are primarily controlled and monitored. Examples include smart appliances,
 3288 sensors, and actuators. Things mostly take the role of Server but they may also take the role of
 3289 Client, for example in machine-to-machine communications.
- 3290 • **User Devices:** Devices employed by the users enabling the users to access resources and
 3291 services. Examples include smart phones, tablets, and wearable devices. User Devices mainly
 3292 take the role of Client, but may also take the role of Server or Intermediary.

- 3293 • **Service Gateways:** Network equipment which take the role of Intermediary. Examples are
3294 home gateways.
- 3295 • **Infra Servers:** Data centers residing in cloud infrastructure, which facilitate the interaction
3296 among Devices by providing network services such as AAA, NAT traversal or discovery. It can
3297 also play the role of Client or Intermediary

DRAFT

3298
3299
3300
3301

Annex C (informative)

Other Resource Models and OCF Mapping

3302

C.1 Multiple resource models

3303 RESTful interactions are defined dependent on the resource model; hence, Devices require a
3304 common understanding of the resource model for interoperability.

3305 There are multiple resource models defined by different organizations including OCF, IPSO
3306 Alliance and oneM2M, and used in the industry, which may restrict interoperability among
3307 respective ecosystems. The main differences from Resource model are as follows:

- 3308 • **Resource structure:** Resources may be defined to have properties (e.g., oneM2M defined
3309 resources), or may be defined as an atomic entity and not be decomposable into properties
3310 (e.g., IPSO alliance defined resources). For example, a smart light may be represented as a
3311 resource with an on-off property or a resource collection containing an on-off resource. In the
3312 former, on-off property doesn't have a URI of its own and can only be accessed indirectly via
3313 the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and
3314 can be directly manipulated.
- 3315 • **Resource name & type:** Resources may be allowed to be named freely and have their
3316 characteristics indicated using a Resource Type property (e.g., as defined in oneM2M).
3317 Alternatively, the name of resources may be defined a priori in a way that the name by itself is
3318 indicative of its characteristic (e.g., as defined by IPSO alliance). For example, in oneM2M
3319 resource model, a smart light can be named with no restrictions, such as 'LivingRoomLight_1"
3320 but in IPSO alliance resource model it is required to have the fixed Object name with numerical
3321 Object ID of "IPSO Light Control (3311)". Consequently, it's likely that in the former case the
3322 data path in URI is freely defined and in the latter case it is predetermined.
- 3323 • **Resource hierarchy:** Resources may be allowed to be organized in hierarchy where a resource
3324 contains another resource with a parent-child relationship (e.g., in oneM2M definition of
3325 resource model). Resources may also be required to have a flat structure and associate with
3326 other resources only by referencing their links.

3327 In addition to the above, different organizations use different syntax and define different features
3328 (e.g., resource interface), which preclude interoperability.

3329

C.2 OCF approach for support of multiple resource models

3330 In order to expand the IoT ecosystem the Framework takes an inclusive approach for interworking
3331 with existing resource models. Specifically, the Framework defines a resource model while
3332 providing a mechanism to easily map to other models. By embracing existing resource models
3333 OCF is inclusive of existing ecosystems while allowing for the transition toward definition of a
3334 comprehensive resource model integrating all ecosystems.

3335 The following OCF characteristics enable support of other resource models:

- 3336 • **resource model is the superset of multiple models:** the resource model is defined as the
3337 superset of existing resource models. In other words, any existing resource model can be
3338 mapped to a subset of resource model concepts.
- 3339 • **Framework may allow for resource model negotiation:** the Client and Server exchange the
3340 information about what resource model(s) each supports. Based on the exchanged information,
3341 the Client and Server choose a resource model to perform RESTful interactions or to perform
3342 translation. This feature is out of scope of the current version of this specification, however,
3343 the following is a high level description for resource model negotiation.

3344 **C.3 Resource model indication**

3345 The Client and server exchange the information about what resource model(s) each supports.
3346 Based on the exchanged information, the Client and Server choose a resource model to perform
3347 RESTful interactions or to perform translation. The exchange could be part of discovery and
3348 negotiation. Based on the exchange, the Client and Server follow a procedure to ensure
3349 interoperability among them. They may choose a common resource model or execute translation
3350 between resource models.

- 3351 • **Resource model schema exchange:** The Client and Server may share the resource model
3352 information when they initiate a RESTful interaction. They may exchange the information about
3353 which resource model they support as part of session establishment procedures. Alternatively,
3354 each request or response message may carry the indication of which resource model it is using.
3355 For example, [COAP] defines “Content-Format option” to indicate the “representation format”
3356 such as “application/json”. It’s possible to extend the Content-Format Option to indicate the
3357 resource model used with the representation format such as “application/ipso-json”.
- 3358 • **Ensuing procedures:** After the Client and Server exchange the resource model information,
3359 they perform a suitable procedure to ensure interoperability among them. The simplest way is
3360 to choose a resource model supported by both the Client and Server. In case there is no
3361 common resource model, the Client and Server may interact through a 3rd party.

3362 In addition to translation which can be resource intensive, a method based on profiles can be used
3363 in which an OCF implementation can accommodate multiple profiles and hence multiple
3364 ecosystems.

- 3365 • **Resource Model Profile:** the Framework defines resource model profiles and implementers or
3366 users choose the active profile. The chosen profile constraints the Device to strict rules in how
3367 resources are defined, instantiated and interacted with. This would allow for interoperability with
3368 devices from the ecosystem identified by the profile (e.g., IPSO, OneM2M etc.). Although this
3369 enables a Device to participate in and be part of any given ecosystem, this scheme does not
3370 allow for generic interoperability at runtime. While this approach may be suitable for resource
3371 constrained devices, more resource capable devices are expected to support more than one
3372 profile.

3373 **C.4 An Example Profile (IPSO profile)**

3374 IPSO defines smart objects that have specific resources and they take values determined by the
3375 data type of that resource. The smart object specification defines a category of such objects. Each
3376 resource represents a characteristic of the smart object being modelled.

3377 While the terms may be different, there are equivalent concepts in OCF to represent these terms.
3378 This section provides the equivalent OCF terms and then frames the IPSO smart object in OCF
3379 terms.

3380 The IPSO object Light Control defined in Section 16 of the IPSO Smart Objects 1.0 is used as the
3381 reference example.

3382 **C.4.1 Conceptual equivalence**

3383 The IPSO smart object definition is equivalent to an Resource Type definition which defines the
3384 relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent to
3385 a Property that like an IPSO Resource has a defined data type, enumeration of acceptable values,
3386 units, a general description and access modes (based on the Interface).

3387 The general method for developing the equivalent Resource Type from an IPSO Smart Object
3388 definition is to ignore the Object ID and replace the Object URN with an OCF ‘.’ (dot) separated
3389 name that incorporates the IPSO object. Alternatively the Object URN can be used as the Resource

3390 Type ID as is (as long as the URN does not contain any '.' (dots)) – using the same Object URN
 3391 as the Resource Type ID allows for compatibility when interacting with an IPSO compliant device.
 3392 The object URN based naming does not have any bearing for OCF to OCF interoperability and so
 3393 the OCF format is preferred – for OCF to OCF interoperability only the data model consistency is
 3394 required.

3395 Two models are available to render IPSO objects into OCF.

3396 1) One is where the IPSO Smart Object represents a Resource. In this case, the IP Smart Object
 3397 is regarded as a resource with the Resource Type matching the description of the Smart Object.
 3398 Furthermore, each resource in the IPSO definition is represented as a Property in the Resource
 3399 Type (the IPSO Resource ID is replaced with a string representing the Property). This is the
 3400 preferred approach when the IPSO Data Model is expressed in the Resource Model.

3401 16) The other approach is to model an IPSO Smart Object as a Collection. Each IPSO
 3402 Resource is then modelled as a Resource with an Resource Type that matches the
 3403 definition of the IPSO Resource. Each of these resource instances are then bound to the
 3404 Collection that represents this IPSO Smart Object.

3405

3406 Below is an example showing how an IPSO LightControl Object is modelled as a Resource.

3407 **Resource Type: Light Control**

3408 Description: This Object is used to control a light source, such as a LED or other light. It allows a
 3409 light to be turned on or off and its dimmer setting to be controlled as a percentage value between
 3410 0 and 100. An optional colour setting enables a string to be used to indicate the desired colour.
 3411 Table 39 and Table 40 define the Resource Type and its properties, respectively.

3412 **Table 39. Light control Resource Type definition**

Resource Type	Resource Type ID	Multiple Instances	Description
Light Control	"oic.light.control" or "urn:oma:lwm2m:ext:3311"	Yes	Light control object with on/off and optional dimming and energy monitor

3413

3414 **Table 40. Light control Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
On/Off	"on-off"	boolean			R, W	yes	On/Of Control: 0 = Off 1 = On
Dimmer	"dim"	integer		%	R, W	no	Proportional Control, integer value between 0 and 100 as percentage
Color	"color"	string	0 – 100	Defined by "units" property	R, W	no	String representing some value in color space
Units	"units"	string			R	no	Measurement Units Definition e.g., "Cel" for Temperature in Celsius.
On Time	"ontime"	integer		s	R, W	no	The time in seconds that the light has been on.

							Writing a value of 0 resets the counter
Cumulative active power	"cumap"	float		Wh	R	no	The cumulative active power since the last cumulative energy reset or device start
Power Factor	"powfact"	float			R	no	The power factor of the load

3415
3416

DRAFT

3417
3418
3419
3420

Annex D (normative)

Resource Type definitions

3421 D.1 List of Resource Type definitions

3422 Table 41 contains the list of defined core resources in this specification.

3423 **Table 41. Alphabetized list of core resources**

Friendly Name (informative)	Resource Type (rt)	Section
Collections	oic.wk.col	D.2
Device Configuration	oic.wk.con	D.3
Platform Configuration	oic.wk.con.p	D.4
Device	oic.wk.d	D.5
Discoverable Resources, baseline interface	oic.wk.res	D.9
Discoverable Resources, link list interface	oic.wk.res	D.10
Icon	oic.r.icon	D.15
Introspection	oic.wk.introspection	D.16
Maintenance	oic.wk.mnt	D.6
Platform	oic.wk.p	D.7
Ping	oic.wk.ping	D.8
Resource Directory	oic.wk.rd	D.14
Scenes (Top Level)	oic.wk.sceneList	D.11
Scenes Collections	oic.wk.sceneCollection	D.12
Scenes Member	oic.wk.sceneMember	D.13

3424

3425

3426 **D.2 OCF Collection**

3427 **D.2.1 Introduction**

3428 OCF Collection Resource Type contains properties and links. The oic.if.baseline interface exposes
3429 a representation of the links and the properties of the collection resource itself

3430 **D.2.2 Example URI**

3431 /CollectionBaselineInterfaceURI

3432 **D.2.3 Resource Type**

3433 The resource type (rt) is defined as: oic.wk.col.

3434 **D.2.4 RAML Definition**

```
3435 #%RAML 0.8
3436 title: Collections
3437 version: 1.0
3438 traits:
3439   - interface-ll :
3440     queryParameters:
3441       if:
3442         enum: ["oic.if.ll"]
3443   - interface-b :
3444     queryParameters:
3445       if:
3446         enum: ["oic.if.b"]
3447   - interface-baseline :
3448     queryParameters:
3449       if:
3450         enum: ["oic.if.baseline"]
3451
3452 /CollectionBaselineInterfaceURI:
3453   description: |
3454     OCF Collection Resource Type contains properties and links.
3455     The oic.if.baseline interface exposes a representation of
3456     the links and the properties of the collection resource itself
3457
3458   is : ['interface-baseline']
3459   get:
3460     description: |
3461       Retrieve on Baseline Interface
3462
3463   responses :
3464     200:
3465       body:
3466         application/json:
3467           schema: /
3468             {
3469               "$schema": "http://json-schema.org/draft-04/schema#",
3470               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3471 reserved.",
3472               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3473 schema.json#",
3474               "title": "Collection",
3475               "definitions": {
3476                 "oic.collection.setoflinks": {
```

```

3477         "description": "A set (array) of simple or individual OIC Links. In
3478 addition to properties required for an OIC Link, the identifier for that link in this set is also
3479 required",
3480         "type": "array",
3481         "items": {
3482             "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3483         }
3484     },
3485     "oic.collection.alllinks": {
3486         "description": "All forms of links in a collection",
3487         "oneOf": [
3488             {
3489                 "$ref": "#/definitions/oic.collection.setoflinks"
3490             }
3491         ]
3492     },
3493     "oic.collection": {
3494         "type": "object",
3495         "description": "A collection is a set (array) of tagged-link or set
3496 (array) of simple links along with additional properties to describe the collection itself",
3497         "properties": {
3498             "id": {
3499                 "anyOf": [
3500                     {
3501                         "type": "integer",
3502                         "description": "A number that is unique to that
3503 collection; like an ordinal number that is not repeated"
3504                     },
3505                     {
3506                         "type": "string",
3507                         "description": "A unique string that could be a hash or
3508 similarly unique"
3509                     },
3510                     {
3511                         "$ref": "oic.types-schema.json#/definitions/uuid",
3512                         "description": "A unique string that could be a UUIDv4"
3513                     }
3514                 ],
3515                 "description": "ID for the collection. Can be an value that is
3516 unique to the use context or a UUIDv4"
3517             },
3518             "di": {
3519                 "$ref": "oic.types-schema.json#/definitions/uuid",
3520                 "description": "The device ID which is an UUIDv4 string; used for
3521 backward compatibility with Spec A definition of /oic/res"
3522             },
3523             "rts": {
3524                 "$ref": "oic.core-
3525 schema.json#/definitions/oic.core/properties/rt",
3526                 "description": "Defines the list of allowable resource types (for
3527 Target and anchors) in links included in the collection; new links being created can only be from
3528 this list"
3529             },
3530             "drel": {
3531                 "type": "string",
3532                 "description": "When specified this is the default relationship
3533 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3534             },
3535             "links": {
3536                 "$ref": "#/definitions/oic.collection.alllinks"
3537             }
3538         }
3539     },
3540     "type": "object",
3541     "allOf": [
3542         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3543         {"$ref": "#/definitions/oic.collection"}
3544     ]
3545 }
3546

```

```

3547     example: /
3548         {
3549             "rt": ["oic.wk.col"],
3550             "id": "unique_example_id",
3551             "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
3552             "links": [
3553                 {
3554                     "href": "switch",
3555                     "rt": [ "oic.r.switch.binary" ],
3556                     "if": [ "oic.if.a", "oic.if.baseline" ],
3557                     "eps": [
3558                         { "ep": "coap://[fe80::b1d6]:1111", "pri": 2 },
3559                         { "ep": "coaps://[fe80::b1d6]:1122" },
3560                         { "ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3 }
3561                     ]
3562                 },
3563                 {
3564                     "href": "airFlow",
3565                     "rt": [ "oic.r.airflow" ],
3566                     "if": [ "oic.if.a", "oic.if.baseline" ],
3567                     "eps": [
3568                         { "ep": "coap://[fe80::b1d6]:1111", "pri": 2 },
3569                         { "ep": "coaps://[fe80::b1d6]:1122" },
3570                         { "ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3 }
3571                     ]
3572                 }
3573             ]
3574         }
3575
3576     post:
3577         description: |
3578             Update on Baseline Interface
3579
3580     body:
3581         application/json:
3582             schema: /
3583                 {
3584                     "$schema": "http://json-schema.org/draft-04/schema#",
3585                     "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3586 reserved.",
3587                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3588 schema.json#",
3589                     "title": "Collection",
3590                     "definitions": {
3591                         "oic.collection.setoflinks": {
3592                             "description": "A set (array) of simple or individual OIC Links. In addition
3593 to properties required for an OIC Link, the identifier for that link in this set is also required",
3594                             "type": "array",
3595                             "items": {
3596                                 "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3597                             }
3598                         },
3599                         "oic.collection.alllinks": {
3600                             "description": "All forms of links in a collection",
3601                             "oneOf": [
3602                                 {
3603                                     "$ref": "#/definitions/oic.collection.setoflinks"
3604                                 }
3605                             ]
3606                         },
3607                         "oic.collection": {
3608                             "type": "object",
3609                             "description": "A collection is a set (array) of tagged-link or set (array)
3610 of simple links along with additional properties to describe the collection itself",
3611                             "properties": {
3612                                 "id": {
3613                                     "anyOf": [
3614                                         {

```

```

3615         "type": "integer",
3616         "description": "A number that is unique to that collection;
3617 like an ordinal number that is not repeated"
3618     },
3619     {
3620         "type": "string",
3621         "description": "A unique string that could be a hash or
3622 similarly unique"
3623     },
3624     {
3625         "$ref": "oic.types-schema.json#/definitions/uuid",
3626         "description": "A unique string that could be a UUIDv4"
3627     }
3628 ],
3629     "description": "ID for the collection. Can be an value that is unique
3630 to the use context or a UUIDv4"
3631 },
3632     "di": {
3633         "$ref": "oic.types-schema.json#/definitions/uuid",
3634         "description": "The device ID which is an UUIDv4 string; used for
3635 backward compatibility with Spec A definition of /oic/res"
3636     },
3637     "rts": {
3638         "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
3639         "description": "Defines the list of allowable resource types (for
3640 Target and anchors) in links included in the collection; new links being created can only be from
3641 this list"
3642     },
3643     "drel": {
3644         "type": "string",
3645         "description": "When specified this is the default relationship to
3646 use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3647     },
3648     "links": {
3649         "$ref": "#/definitions/oic.collection.alllinks"
3650     }
3651 },
3652 },
3653     "type": "object",
3654     "allOf": [
3655         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3656         {"$ref": "#/definitions/oic.collection"}
3657     ]
3658 }
3659
3660 responses :
3661     200:
3662         body:
3663             application/json:
3664                 schema: /
3665                     {
3666                         "$schema": "http://json-schema.org/draft-04/schema#",
3667                         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3668 reserved.",
3669                         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3670 schema.json#",
3671                         "title": "Collection",
3672                         "definitions": {
3673                             "oic.collection.setoflinks": {
3674                                 "description": "A set (array) of simple or individual OIC Links. In
3675 addition to properties required for an OIC Link, the identifier for that link in this set is also
3676 required",
3677                                 "type": "array",
3678                                 "items": {
3679                                     "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3680                                 }
3681                             },
3682                             "oic.collection.alllinks": {

```

```

3683         "description": "All forms of links in a collection",
3684         "oneOf": [
3685             {
3686                 "$ref": "#/definitions/oic.collection.setoflinks"
3687             }
3688         ]
3689     },
3690     "oic.collection": {
3691         "type": "object",
3692         "description": "A collection is a set (array) of tagged-link or set
3693 (array) of simple links along with additional properties to describe the collection itself",
3694         "properties": {
3695             "id": {
3696                 "anyOf": [
3697                     {
3698                         "type": "integer",
3699                         "description": "A number that is unique to that
3700 collection; like an ordinal number that is not repeated"
3701                     },
3702                     {
3703                         "type": "string",
3704                         "description": "A unique string that could be a hash or
3705 similarly unique"
3706                     },
3707                     {
3708                         "$ref": "oic.types-schema.json#/definitions/uuid",
3709                         "description": "A unique string that could be a UUIDv4"
3710                     }
3711                 ],
3712                 "description": "ID for the collection. Can be a value that is
3713 unique to the use context or a UUIDv4"
3714             },
3715             "di": {
3716                 "$ref": "oic.types-schema.json#/definitions/uuid",
3717                 "description": "The device ID which is an UUIDv4 string; used for
3718 backward compatibility with Spec A definition of /oic/res"
3719             },
3720             "rts": {
3721                 "$ref": "oic.core-
3722 schema.json#/definitions/oic.core/properties/rt",
3723                 "description": "Defines the list of allowable resource types (for
3724 Target and anchors) in links included in the collection; new links being created can only be from
3725 this list"
3726             },
3727             "drel": {
3728                 "type": "string",
3729                 "description": "When specified this is the default relationship
3730 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3731             },
3732             "links": {
3733                 "$ref": "#/definitions/oic.collection.alllinks"
3734             }
3735         }
3736     },
3737     "type": "object",
3738     "allOf": [
3739         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3740         {"$ref": "#/definitions/oic.collection"}
3741     ]
3742 }
3743

```

3744 D.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	yes	Read Write	Resource Type
di	multiple types: see schema		Read Write	Unique identifier for device (UUID)

title	string		Read Write	A title for the link relation. Can be used by the UI to provide a context
eps	array: see schema	yes	Read Write	the Endpoint information of the target Resource
ins	multiple types: see schema		Read Write	The instance identifier for this web link in an array of web links - used in collections
p	object: see schema		Read Write	Specifies the framework policies on the Resource referenced by the target URI
href	string	yes	Read Write	This is the target URI, it can be specified as a Relative Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique.
rel	multiple types: see schema		Read Write	The relation of the target URI referenced by the link to the context URI
type	array: see schema		Read Write	A hint at the representation of the resource referenced by the target URI. This represents the media types that are used for both accepting and emitting
anchor	string		Read Write	This is used to override the context URI e.g. override the URI of the containing collection

if	array: schema	see	yes	Read Write	The interface set supported by this resource
----	------------------	-----	-----	------------	--

3745 **D.2.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/CollectionBaselineInterfaceURI		get	post		

3746 **D.2.7 Referenced JSON schemas**

3747 **D.2.8 oic.oic-link-schema.json**

```

3748 {
3749   "$schema": "http://json-schema.org/draft-04/schema#",
3750   "description": "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All rights
3751 reserved.",
3752   "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
3753   "definitions": {
3754     "oic.oic-link": {
3755       "type": "object",
3756       "properties": {
3757         "href": {
3758           "type": "string",
3759           "maxLength": 256,
3760           "description": "This is the target URI, it can be specified as a Relative Reference or
3761 fully-qualified URI. Relative Reference should be used along with the di parameter to make it
3762 unique.",
3763           "format": "uri"
3764         },
3765         "rel": {
3766           "oneOf": [
3767             {
3768               "type": "array",
3769               "items": {
3770                 "type": "string",
3771                 "maxLength": 64
3772               },
3773               "minItems": 1,
3774               "default": ["hosts"]
3775             },
3776             {
3777               "type": "string",
3778               "maxLength": 64,
3779               "default": "hosts"
3780             }
3781           ],
3782           "description": "The relation of the target URI referenced by the link to the context URI"
3783         },
3784         "rt": {
3785           "type": "array",
3786           "items": {
3787             "type": "string",
3788             "maxLength": 64
3789           },
3790           "minItems": 1,
3791           "description": "Resource Type"
3792         },
3793         "if": {
3794           "type": "array",
3795           "items": {
3796             "type": "string",
3797             "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
3798 "oic.if.a", "oic.if.s"]
3799           },
3800           "minItems": 1,
3801           "description": "The interface set supported by this resource"
3802         },
3803         "di": {
3804           "$ref": "oic.types-schema.json#/definitions/uuid",
3805           "description": "Unique identifier for device (UUID)"

```

```

3806     },
3807     "p": {
3808         "description": "Specifies the framework policies on the Resource referenced by the target
3809 URI",
3810         "type": "object",
3811         "properties": {
3812             "bm": {
3813                 "description": "Specifies the framework policies on the Resource referenced by the
3814 target URI for e.g. observable and discoverable",
3815                 "type": "integer"
3816             }
3817         },
3818         "required" : ["bm"]
3819     },
3820     "title": {
3821         "type": "string",
3822         "maxLength": 64,
3823         "description": "A title for the link relation. Can be used by the UI to provide a
3824 context"
3825     },
3826     "anchor": {
3827         "type": "string",
3828         "maxLength": 256,
3829         "description": "This is used to override the context URI e.g. override the URI of the
3830 containing collection",
3831         "format": "uri"
3832     },
3833     "ins": {
3834         "oneOf": [
3835             {
3836                 "type": "integer",
3837                 "description": "An ordinal number that is not repeated - must be unique in the
3838 collection context"
3839             },
3840             {
3841                 "type": "string",
3842                 "maxLength": 256,
3843                 "format" : "uri",
3844                 "description": "Any unique string including a URI"
3845             },
3846             {
3847                 "$ref": "oic.types-schema.json#/definitions/uuid",
3848                 "description": "Unique identifier (UUID)"
3849             }
3850         ],
3851         "description": "The instance identifier for this web link in an array of web links - used
3852 in collections"
3853     },
3854     "type": {
3855         "type": "array",
3856         "description": "A hint at the representation of the resource referenced by the target
3857 URI. This represents the media types that are used for both accepting and emitting",
3858         "items" : {
3859             "type": "string",
3860             "maxLength": 64
3861         },
3862         "minItems": 1,
3863         "default": "application/cbor"
3864     },
3865     "eps": {
3866         "type": "array",
3867         "description": "the Endpoint information of the target Resource",
3868         "items": {
3869             "type": "object",
3870             "properties": {
3871                 "ep": {
3872                     "type": "string",
3873                     "format": "uri",
3874                     "description": "URI with Transport Protocol Suites + Endpoint Locator as specified
3875 in 10.2.1"
3876                 }

```

```

3877         "pri": {
3878             "type": "integer",
3879             "minimum": 1,
3880             "description": "The priority among multiple Endpoints as specified in 10.2.3"
3881         }
3882     }
3883 }
3884 }
3885 },
3886 "required": [ "href", "rt", "if", "eps" ]
3887 }
3888 },
3889 "type": "object",
3890 "allOf": [
3891     { "$ref": "#/definitions/oic.oic-link" }
3892 ]
3893 }
3894

```

3895 **D.3 OIC Device Configuration**

3896 **D.3.1 Introduction**

3897 Resource that allows for Device specific information to be configured.

3898 **D.3.2 Example URI**

3899 /example/oic/con

3900 **D.3.3 Resource Type**

3901 The resource type (rt) is defined as: oic.wk.con.

3902 **D.3.4 RAML Definition**

```

3903 #%RAML 0.8
3904 title: OIC Configuration
3905 version: v1-20160622
3906 traits:
3907   - interface-rw :
3908       queryParameters:
3909           if:
3910               enum: ["oic.if.rw"]
3911   - interface-all :
3912       queryParameters:
3913           if:
3914               enum: ["oic.if.rw", "oic.if.baseline"]
3915
3916 /example/oic/con:
3917     description: |
3918         Resource that allows for Device specific information to be configured.
3919
3920     get:
3921         description: |
3922             Retrieves the current Device configuration settings
3923
3924         is : ['interface-all']
3925     responses :
3926         200:
3927             body:
3928                 application/json:
3929                 schema: /

```

```

3930     {
3931         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-
3932 schema.json#",
3933         "$schema": "http://json-schema.org/draft-04/schema#",
3934         "description": "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
3935 rights reserved.",
3936         "definitions": {
3937             "oic.wk.con": {
3938                 "type": "object",
3939                 "properties": {
3940                     "loc": {
3941                         "type": "array",
3942                         "description": "Location information",
3943                         "items": {
3944                             "type": "number"
3945                         },
3946                         "minItems": 2,
3947                         "maxItems": 2
3948                     },
3949                     "locn": {
3950                         "type": "string",
3951                         "maxLength": 64,
3952                         "description": "Human Friendly Name for location"
3953                     },
3954                     "c": {
3955                         "type": "string",
3956                         "maxLength": 64,
3957                         "description": "Currency"
3958                     },
3959                     "r": {
3960                         "type": "string",
3961                         "maxLength": 64,
3962                         "description": "Region"
3963                     },
3964                     "ln": {
3965                         "type": "array",
3966                         "items": :
3967                         {
3968                             "type": "object",
3969                             "properties": {
3970                                 "language": {
3971                                     "$ref": "oic.types-schema.json#/definitions/language-tag",
3972                                     "description": "An RFC 5646 language tag."
3973                                 },
3974                                 "value": {
3975                                     "type": "string",
3976                                     "maxLength": 64,
3977                                     "description": "Device description in the indicated language."
3978                                 }
3979                             }
3980                         },
3981                     "minItems" : 1,
3982                     "description": "Localized names"
3983                 },
3984                 "dl": {
3985                     "$ref": "oic.types-schema.json#/definitions/language-tag",
3986                     "description": "Default Language"
3987                 }
3988             }
3989         },
3990     },
3991     "type": "object",
3992     "allOf": [
3993         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
3994         { "$ref": "#/definitions/oic.wk.con" }
3995     ],
3996     "required": ["n"]
3997 }
3998
3999     example: /

```

```

4000     {
4001         "n": "My Friendly Device Name",
4002         "rt": ["oic.wk.con"],
4003         "loc": [32.777,-96.797],
4004         "locn": "My Location Name",
4005         "c": "USD",
4006         "r": "MyRegion",
4007         "dl": "en"
4008     }
4009
4010     post:
4011         description: |
4012             Update the information about the Device
4013
4014         is : ['interface-rw']
4015         body:
4016             application/json:
4017                 schema: /
4018                     {
4019                         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-Update-
4020 schema.json#",
4021                         "$schema": "http://json-schema.org/draft-04/schema#",
4022                         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4023 reserved.",
4024                         "definitions": {
4025                             "oic.wk.con": {
4026                                 "type": "object",
4027                                 "properties": {
4028                                     "loc": {
4029                                         "type": "array",
4030                                         "description": "Location information",
4031                                         "items": {
4032                                             "type": "number"
4033                                         },
4034                                         "minItems": 2,
4035                                         "maxItems": 2
4036                                     },
4037                                     "locn": {
4038                                         "type": "string",
4039                                         "maxLength": 64,
4040                                         "description": "Human Friendly Name for location"
4041                                     },
4042                                     "c": {
4043                                         "type": "string",
4044                                         "maxLength": 64,
4045                                         "description": "Currency"
4046                                     },
4047                                     "r": {
4048                                         "type": "string",
4049                                         "maxLength": 64,
4050                                         "description": "Region"
4051                                     },
4052                                     "ln": {
4053                                         "type": "array",
4054                                         "items" :
4055                                             {
4056                                                 "type": "object",
4057                                                 "properties": {
4058                                                     "language": {
4059                                                         "$ref": "oic.types-schema.json#/definitions/language-tag",
4060                                                         "description": "An RFC 5646 language tag."
4061                                                     },
4062                                                     "value": {
4063                                                         "type": "string",
4064                                                         "maxLength": 64,
4065                                                         "description": "Device description in the indicated language."
4066                                                     }
4067                                                 }
4068                                             }
4069                                     }
4070                                 }
4071                             }
4072                         }
4073                     }

```

```

4068         },
4069         "minItems" : 1,
4070         "description": "Localized names"
4071     },
4072     "dl": {
4073         "$ref": "oic.types-schema.json#/definitions/language-tag",
4074         "description": "Default Language"
4075     }
4076 }
4077 }
4078 },
4079 "type": "object",
4080 "allOf": [
4081     { "$ref": "oic.core-schema.rw.json#/definitions/oic.core" },
4082     { "$ref": "#/definitions/oic.wk.con" }
4083 ],
4084 "required": ["n"]
4085 }
4086
4087 example: /
4088 {
4089     "n": "Nuevo Nombre Amistoso",
4090     "r": "MyNewRegion",
4091     "ln": [ { "language": "es", "value": "Nuevo Nombre Amistoso" } ],
4092     "dl": "es"
4093 }
4094
4095 responses :
4096     200:
4097         body:
4098             application/json:
4099                 schema: /
4100                 {
4101                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-Update-
4102 schema.json#",
4103                     "$schema": "http://json-schema.org/draft-04/schema#",
4104                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4105 reserved.",
4106                     "definitions": {
4107                         "oic.wk.con": {
4108                             "type": "object",
4109                             "properties": {
4110                                 "loc": {
4111                                     "type": "array",
4112                                     "description": "Location information",
4113                                     "items": {
4114                                         "type": "number"
4115                                     },
4116                                     "minItems": 2,
4117                                     "maxItems": 2
4118                                 },
4119                                 "locn": {
4120                                     "type": "string",
4121                                     "maxLength": 64,
4122                                     "description": "Human Friendly Name for location"
4123                                 },
4124                                 "c": {
4125                                     "type": "string",
4126                                     "maxLength": 64,
4127                                     "description": "Currency"
4128                                 },
4129                                 "r": {
4130                                     "type": "string",
4131                                     "maxLength": 64,
4132                                     "description": "Region"
4133                                 },
4134                                 "ln": {

```

```

4135         "type": "array",
4136         "items" :
4137         {
4138             "type": "object",
4139             "properties": {
4140                 "language": {
4141                     "$ref": "oic.types-schema.json#/definitions/language-tag",
4142                     "description": "An RFC 5646 language tag."
4143                 },
4144                 "value": {
4145                     "type": "string",
4146                     "maxLength": 64,
4147                     "description": "Device description in the indicated language."
4148                 }
4149             }
4150         },
4151         "minItems" : 1,
4152         "description": "Localized names"
4153     },
4154     "dl": {
4155         "$ref": "oic.types-schema.json#/definitions/language-tag",
4156         "description": "Default Language"
4157     }
4158 }
4159 },
4160 },
4161 "type": "object",
4162 "allOf": [
4163     { "$ref": "oic.core-schema.rw.json#/definitions/oic.core"},
4164     { "$ref": "#/definitions/oic.wk.con" }
4165 ],
4166 "required": ["n"]
4167 }
4168
4169 example: /
4170 {
4171     "n": "Nuevo Nombre Amistoso",
4172     "r": "MyNewRegion",
4173     "ln": [ { "language": "es", "value": "Nuevo Nombre Amistoso" } ],
4174     "dl": "es",
4175     "mnpn": [ { "language": "es", "value": "Nuevo nombre de Plataforma Amigable" } ]
4176 }
4177

```

4178 D.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
loc	array: see schema		Read Write	Location information
c	string		Read Write	Currency
ln	array: see schema		Read Write	Localized names
locn	string		Read Write	Human Friendly Name for location
dl	multiple types: see schema		Read Write	Default Language
r	string		Read Write	Region

4179 D.3.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/example/oic/con		get	post		

4180 D.4 OIC Platform Configuration

4181 D.4.1 Introduction

4182 Resource that allows for platform specific information to be configured.

4183 D.4.2 Example URI

4184 /example/oic/con/p

4185 D.4.3 Resource Type

4186 The resource type (rt) is defined as: oic.wk.con.p.

4187 D.4.4 RAML Definition

```
4188 #%RAML 0.8
4189 title: OIC Configuration
4190 version: v1-20160622
4191 traits:
4192   - interface-rw :
4193     queryParameters:
4194       if:
4195         enum: ["oic.if.rw"]
4196   - interface-all :
4197     queryParameters:
4198       if:
4199         enum: ["oic.if.rw", "oic.if.baseline"]
4200
4201 /example/oic/con/p:
4202   description: |
4203     Resource that allows for platform specific information to be configured.
4204
4205   get:
4206     description: |
4207       Retrieves the current platform configuration settings
4208
4209     is : ['interface-all']
4210     responses :
4211       200:
4212         body:
4213           application/json:
4214             schema: /
4215               {
4216                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con.p-
4217 schema.json#",
4218                 "$schema": "http://json-schema.org/draft-04/schema#",
4219                 "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
4220 reserved.",
4221                 "definitions": {
4222                   "oic.wk.con.p": {
4223                     "type": "object",
4224                     "properties": {
4225                       "mnpn": {
4226                         "type": "array",
4227                         "items" :
4228                           {
4229                             "type": "object",
4230                             "properties": {
4231                               "language": {
4232                                 "$ref": "oic.types-schema.json#/definitions/language-tag",
4233                                 "description": "An RFC 5646 language tag."
4234                               }

```

```

4235         "value": {
4236             "type": "string",
4237             "maxLength": 64,
4238             "description": "Platform description in the indicated language."
4239         }
4240     },
4241 },
4242     "minItems" : 1,
4243     "description": "Platform names"
4244 }
4245 }
4246 }
4247 },
4248 "type": "object",
4249 "allOf": [
4250     { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4251     { "$ref": "#/definitions/oic.wk.con.p" }
4252 ]
4253 }
4254
4255 example: /
4256 {
4257     "rt": [ "oic.wk.con.p"],
4258     "mnpn": [ { "language": "en", "value": "My Friendly Device Name" } ]
4259 }
4260
4261 post:
4262     description: |
4263         Update the information about the platform
4264
4265     is : ['interface-rw']
4266     body:
4267         application/json:
4268             schema: /
4269                 {
4270                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con.p-Update-
4271 schema.json#",
4272                     "$schema": "http://json-schema.org/draft-04/schema#",
4273                     "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
4274 reserved.",
4275                     "definitions": {
4276                         "oic.wk.con.p": {
4277                             "type": "object",
4278                             "properties": {
4279                                 "mnpn": {
4280                                     "type": "array",
4281                                     "items" :
4282                                         {
4283                                             "type": "object",
4284                                             "properties": {
4285                                                 "language": {
4286                                                     "$ref": "oic.types-schema.json#/definitions/language-tag",
4287                                                     "description": "An RFC 5646 language tag."
4288                                                 },
4289                                                 "value": {
4290                                                     "type": "string",
4291                                                     "maxLength": 64,
4292                                                     "description": "Platform description in the indicated language."
4293                                                 }
4294                                             }
4295                                         },
4296                                     "minItems" : 1,
4297                                     "description": "Platform names"
4298                                 }
4299                             }
4300                         }
4301 },

```

```

4302         "type": "object",
4303         "allOf": [
4304             { "$ref": "oic.core-schema.rw.json#/definitions/oic.core"},
4305             { "$ref": "#/definitions/oic.wk.con.p" }
4306         ],
4307         "required": ["mnpn"]
4308     }
4309
4310     example: /
4311     {
4312         "n": "Nuevo nombre",
4313         "mnpn": [ { "language": "es", "value": "Nuevo nombre de Plataforma Amigable" } ]
4314     }
4315
4316     responses :
4317     200:
4318         body:
4319         application/json:
4320             schema: /
4321             {
4322                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con.p-Update-
4323 schema.json#",
4324                 "$schema": "http://json-schema.org/draft-04/schema#",
4325                 "description" : "Copyright (c) 2017 Open Connectivity Foundation, Inc. All rights
4326 reserved.",
4327                 "definitions": {
4328                     "oic.wk.con.p": {
4329                         "type": "object",
4330                         "properties": {
4331                             "mnpn": {
4332                                 "type": "array",
4333                                 "items" :
4334                                 {
4335                                     "type": "object",
4336                                     "properties": {
4337                                         "language": {
4338                                             "$ref": "oic.types-schema.json#/definitions/language-tag",
4339                                             "description": "An RFC 5646 language tag."
4340                                         },
4341                                         "value": {
4342                                             "type": "string",
4343                                             "maxLength": 64,
4344                                             "description": "Platform description in the indicated language."
4345                                         }
4346                                     }
4347                                 },
4348                                 "minItems" : 1,
4349                                 "description": "Platform names"
4350                             }
4351                         }
4352                     },
4353                 },
4354                 "type": "object",
4355                 "allOf": [
4356                     { "$ref": "oic.core-schema.rw.json#/definitions/oic.core"},
4357                     { "$ref": "#/definitions/oic.wk.con.p" }
4358                 ],
4359                 "required": ["mnpn"]
4360             }
4361
4362     example: /
4363     {
4364         "n": "Nuevo nombre",
4365         "mnpn": [ { "language": "es", "value": "Nuevo nombre de Plataforma Amigable" } ]
4366     }
4367

```

4368 **D.4.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
mnpn	array: see schema		Read Write	Platform names

4369 **D.4.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/example/oic/con/p		get	post		

4370 **D.5 Device**

4371 **D.5.1 Introduction**

4372 Known resource that is hosted by every Server. Allows for logical device specific information to be
4373 discovered.

4374 **D.5.2 Wellknown URI**

4375 /oic/d

4376 **D.5.3 Resource Type**

4377 The resource type (rt) is defined as: oic.wk.d.

4378 **D.5.4 RAML Definition**

```

4379 #%RAML 0.8
4380 title: OIC Root Device
4381 version: v1-20160622
4382 traits:
4383   - interface :
4384     queryParameters:
4385       if:
4386         enum: ["oic.if.r", "oic.if.baseline"]
4387
4388 /oic/d:
4389   description: |
4390     Known resource that is hosted by every Server.
4391     Allows for logical device specific information to be discovered.
4392
4393   is : ['interface']
4394   get:
4395     description: |
4396       Retrieve the information about the Device
4397
4398   responses :
4399     200:
4400       body:
4401         application/json:
4402           schema: /
4403             {
4404               "$schema": "http://json-schemas.org/draft-04/schema#",
4405               "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
4406 rights reserved.",
4407               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.d-
4408 schema.json#",
4409               "definitions": {
4410                 "oic.wk.d": {
4411                   "type": "object",
4412                   "properties": {
4413                     "di": {

```

```

4414         "$ref": "oic.types-schema.json#/definitions/uuid",
4415         "readOnly": true,
4416         "description": "Unique identifier for device (UUID)"
4417     },
4418     "icv": {
4419         "type": "string",
4420         "maxLength": 64,
4421         "readOnly": true,
4422         "description": "The version of the OIC Server"
4423     },
4424     "dmv": {
4425         "type": "string",
4426         "maxLength": 256,
4427         "readOnly": true,
4428         "description": "Spec versions of the Resource and Device Specifications to
4429 which this device data model is implemented"
4430     },
4431     "ld": {
4432         "type": "array",
4433         "items" :
4434         {
4435             "type": "object",
4436             "properties": {
4437                 "language": {
4438                     "$ref": "oic.types-schema.json#/definitions/language-tag",
4439                     "readOnly": true,
4440                     "description": "An RFC 5646 language tag."
4441                 },
4442                 "value": {
4443                     "type": "string",
4444                     "maxLength": 64,
4445                     "readOnly": true,
4446                     "description": "Device description in the indicated language."
4447                 }
4448             }
4449         },
4450         "minItems" : 1,
4451         "readOnly": true,
4452         "description": "Localized Description."
4453     },
4454     "sv": {
4455         "type": "string",
4456         "maxLength": 64,
4457         "readOnly": true,
4458         "description": "Software version."
4459     },
4460     "dmn": {
4461         "type": "array",
4462         "items" :
4463         {
4464             "type": "object",
4465             "properties": {
4466                 "language": {
4467                     "$ref": "oic.types-schema.json#/definitions/language-tag",
4468                     "readOnly": true,
4469                     "description": "An RFC 5646 language tag."
4470                 },
4471                 "value": {
4472                     "type": "string",
4473                     "maxLength": 64,
4474                     "readOnly": true,
4475                     "description": "Device description in the indicated language."
4476                 }
4477             }
4478         },
4479         "minItems" : 1,
4480         "readOnly": true,
4481         "description": "Localized Description."
4482     },
4483     "dmno": {
4484         "type": "string",

```

```

4485         "maxLength": 64,
4486         "readOnly": true,
4487         "description": "Model number as designated by manufacturer."
4488     },
4489     "piid": {
4490         "$ref": "oic.types-schema.json#/definitions/uuid",
4491         "readOnly": true,
4492         "description": "Protocol independent unique identifier for device (UUID)
4493 that is immutable."
4494     }
4495 }
4496 }
4497 },
4498 "type": "object",
4499 "allOf": [
4500     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4501     { "$ref": "#/definitions/oic.wk.d" }
4502 ],
4503 "required": [ "n", "di", "icv", "dmv", "piid" ]
4504 }
4505
4506 example: /
4507 {
4508     "n": "Device 1",
4509     "rt": [ "oic.wk.d" ],
4510     "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
4511     "icv": "ocf.1.0.0",
4512     "dmv": "ocf.res.1.0.0, ocf.sh.1.0.0",
4513     "piid": "6F0AAC04-2BB0-468D-B57C-16570A26AE48"
4514 }
4515

```

D.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
ld	array: see schema		Read Only	Localized Description.
piid	multiple types: see schema	yes	Read Only	Protocol independent unique identifier for device (UUID) that is immutable.
di	multiple types: see schema	yes	Read Only	Unique identifier for device (UUID)
dmno	string		Read Only	Model number as designated by manufacturer.
sv	string		Read Only	Software version.
dmn	array: see schema		Read Only	Localized Description.
dmv	string	yes	Read Only	Spec versions of the Resource and Device Specifications to which this device data model is implemented
icv	string	yes	Read Only	The version of the OIC Server

4517 **D.5.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/d		get			

4518 **D.6 Maintenance**

4519 **D.6.1 Introduction**

4520 The resource through which a Device is maintained and can be used for diagnostic purposes. fr
 4521 (Factory Reset) is a boolean. The value 0 means No action (Default), the value 1 means Start
 4522 Factory Reset After factory reset, this value shall be changed back to the default value rb (Reboot)
 4523 is a boolean. The value 0 means No action (Default), the value 1 means Start Reboot After Reboot,
 4524 this value shall be changed back to the default value

4525 **D.6.2 Wellknown URI**

4526 /oic/mnt

4527 **D.6.3 Resource Type**

4528 The resource type (rt) is defined as: oic.wk.mnt.

4529 **D.6.4 RAML Definition**

```

4530 #%RAML 0.8
4531 title: Maintenance
4532 version: v1-20160622
4533 traits:
4534   - interface :
4535     queryParameters:
4536       if:
4537         enum: ["oic.if.rw", "oic.if.r", "oic.if.baseline"]
4538
4539 /oic/mnt:
4540   description: |
4541     The resource through which a Device is maintained and can be used for diagnostic purposes.
4542     fr (Factory Reset) is a boolean.
4543     The value 0 means No action (Default), the value 1 means Start Factory Reset
4544     After factory reset, this value shall be changed back to the default value
4545     rb (Reboot) is a boolean.
4546     The value 0 means No action (Default), the value 1 means Start Reboot
4547     After Reboot, this value shall be changed back to the default value
4548
4549   is : ['interface']
4550   get:
4551     description: |
4552       Retrieve the maintenance action status
4553
4554     queryParameters:
4555       if:
4556         enum: oic.if.rwoic.if.roic.if.baseline
4557     responses :
4558       200:
4559         body:
4560           application/json:
4561             schema: /
4562             {
4563               "$schema": "http://json-schemas.org/draft-04/schema#",
4564               "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
4565               rights reserved.",
  
```

```

4566         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
4567 schema.json#",
4568         "definitions": {
4569             "oic.wk.mnt": {
4570                 "type": "object",
4571                 "anyOf": [
4572                     {"required": ["fr"]},
4573                     {"required": ["rb"]}
4574                 ],
4575                 "properties": {
4576                     "fr": {
4577                         "type": "boolean",
4578                         "description": "Factory Reset"
4579                     },
4580                     "rb": {
4581                         "type": "boolean",
4582                         "description": "Reboot Action"
4583                     }
4584                 }
4585             }
4586         },
4587         "type": "object",
4588         "allOf": [
4589             {"$ref": "oic.core-schema.json#/definitions/oic.core"},
4590             {"$ref": "#/definitions/oic.wk.mnt" }
4591         ]
4592     }
4593
4594     example: /
4595     {
4596         "rt": ["oic.wk.mnt"],
4597         "fr": false,
4598         "rb": false
4599     }
4600
4601     post:
4602         description: |
4603             Set the maintenance action(s)
4604
4605     queryParameters:
4606         if:
4607             enum: oic.if.rwoic.if.baseline
4608
4609     body:
4610         application/json:
4611             schema: /
4612             {
4613                 "$schema": "http://json-schemas.org/draft-04/schema#",
4614                 "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All rights
4615 reserved.",
4616                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-schema.json#",
4617                 "definitions": {
4618                     "oic.wk.mnt": {
4619                         "type": "object",
4620                         "anyOf": [
4621                             {"required": ["fr"]},
4622                             {"required": ["rb"]}
4623                         ],
4624                         "properties": {
4625                             "fr": {
4626                                 "type": "boolean",
4627                                 "description": "Factory Reset"
4628                             },
4629                             "rb": {
4630                                 "type": "boolean",
4631                                 "description": "Reboot Action"
4632                             }
4633                         }
4634                     }
4635                 }

```

```

4632     }
4633     }
4634   },
4635   "type": "object",
4636   "allOf": [
4637     { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4638     { "$ref": "#/definitions/oic.wk.mnt" }
4639   ]
4640 }
4641
4642 example: /
4643 {
4644   "fr": false,
4645   "rb": false
4646 }
4647
4648 responses :
4649 200:
4650   body:
4651     application/json:
4652       schema: /
4653         {
4654           "$schema": "http://json-schemas.org/draft-04/schema#",
4655           "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
4656 rights reserved.",
4657           "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
4658 schema.json#",
4659           "definitions": {
4660             "oic.wk.mnt": {
4661               "type": "object",
4662               "anyOf": [
4663                 {"required": ["fr"]},
4664                 {"required": ["rb"]}
4665               ],
4666               "properties": {
4667                 "fr": {
4668                   "type": "boolean",
4669                   "description": "Factory Reset"
4670                 },
4671                 "rb": {
4672                   "type": "boolean",
4673                   "description": "Reboot Action"
4674                 }
4675               }
4676             }
4677           },
4678           "type": "object",
4679           "allOf": [
4680             { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4681             { "$ref": "#/definitions/oic.wk.mnt" }
4682           ]
4683         }
4684
4685 example: /
4686 {
4687   "fr": false,
4688   "rb": false
4689 }
4690

```

4691 D.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
fr	boolean	yes	Read Write	Factory Reset
rb	boolean	yes	Read Write	Reboot Action

4692 **D.6.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/mnt		get	post		

4693 **D.7 Platform**

4694 **D.7.1 Introduction**

4695 Known resource that is defines the platform on which an Server is hosted. Allows for platform
4696 specific information to be discovered.

4697 **D.7.2 Wellknown URI**

4698 /oic/p

4699 **D.7.3 Resource Type**

4700 The resource type (rt) is defined as: oic.wk.p.

4701 **D.7.4 RAML Definition**

```
4702 #%RAML 0.8
4703 title: Platform
4704 version: v1-20160622
4705 traits:
4706   - interface :
4707     queryParameters:
4708       if:
4709         enum: ["oic.if.r", "oic.if.baseline"]
4710
4711 /oic/p:
4712   description: |
4713     Known resource that is defines the platform on which an Server is hosted.
4714     Allows for platform specific information to be discovered.
4715
4716   is : ['interface']
4717   get:
4718     description: |
4719       Retrieve the information about the Platform
4720
4721   responses :
4722     200:
4723       body:
4724         application/json:
4725           schema: /
4726           {
4727             "$schema": "http://json-schemas.org/draft-04/schema#",
4728             "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
rights reserved.",
4729             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.p-
schema.json#",
4730             "definitions": {
4731               "oic.wk.p": {
4732                 "type": "object",
4733                 "properties": {
4734                   "pi": {
4735                     "$ref": "oic.types-schema.json#/definitions/uuid",
4736                     "readOnly": true,
4737                     "description": "Platform Identifier as a UUID"
4738                   },
4739                   "mnmn": {
4740                     "type": "string",
4741
4742
```

```

4743         "readOnly": true,
4744         "description": "Manufacturer Name",
4745         "maxLength": 64
4746     },
4747     "mnml": {
4748         "type": "string",
4749         "readOnly": true,
4750         "description": "Manufacturer's URL",
4751         "maxLength": 256,
4752         "format": "uri"
4753     },
4754     "mnmo": {
4755         "type": "string",
4756         "maxLength": 64,
4757         "readOnly": true,
4758         "description": "Model number as designated by manufacturer"
4759     },
4760     "mndt": {
4761         "$ref": "oic.types-schema.json#/definitions/date",
4762         "readOnly": true,
4763         "description": "Manufacturing Date."
4764     },
4765     "mnpv": {
4766         "type": "string",
4767         "maxLength": 64,
4768         "readOnly": true,
4769         "description": "Platform Version"
4770     },
4771     "mnos": {
4772         "type": "string",
4773         "maxLength": 64,
4774         "readOnly": true,
4775         "description": "Platform Resident OS Version"
4776     },
4777     "mnhw": {
4778         "type": "string",
4779         "maxLength": 64,
4780         "readOnly": true,
4781         "description": "Platform Hardware Version"
4782     },
4783     "mnfv": {
4784         "type": "string",
4785         "maxLength": 64,
4786         "readOnly": true,
4787         "description": "Manufacturer's firmware version"
4788     },
4789     "mnsl": {
4790         "type": "string",
4791         "readOnly": true,
4792         "description": "Manufacturer's Support Information URL",
4793         "maxLength": 256,
4794         "format": "uri"
4795     },
4796     "st": {
4797         "type": "string",
4798         "readOnly": true,
4799         "description": "Reference time for the device as defined in ISO 8601, where
4800 concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'.",
4801         "format": "date-time"
4802     },
4803     "vid": {
4804         "type": "string",
4805         "maxLength": 64,
4806         "readOnly": true,
4807         "description": "Manufacturer's defined string for the platform. The string
4808 is freeform and up to the manufacturer on what text to populate it"
4809     }
4810 }
4811 },
4812 },
4813 "type": "object",

```

```

4814     "allOf": [
4815         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4816         { "$ref": "#/definitions/oic.wk.p" }
4817     ],
4818     "required": [ "pi", "mnmn" ]
4819 }
4820

```

```

4821 example: /
4822 {
4823     "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
4824     "rt": [ "oic.wk.p" ],
4825     "mnmn": "Acme, Inc"
4826 }
4827

```

4828 D.7.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
mnfv	string		Read Only	Manufacturer's firmware version
vid	string		Read Only	Manufacturer's defined string for the platform. The string is freeform and up to the manufacturer on what text to populate it
mnmn	string	yes	Read Only	Manufacturer Name
mnmo	string		Read Only	Model number as designated by manufacturer
mnml	string		Read Only	Manufacturer's URL
mnos	string		Read Only	Platform Resident OS Version
mndt	multiple types: see schema		Read Only	Manufacturing Date.
st	string		Read Only	Reference time for the device as defined in ISO 8601, where concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'.
mnsi	string		Read Only	Manufacturer's Support Information URL
mpv	string		Read Only	Platform Version
pi	multiple types: see schema	yes	Read Only	Platform Identifier as a UUID

mnhw	string		Read Only	Platform Hardware Version
------	--------	--	-----------	---------------------------

4829 **D.7.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/p		get			

4830 **D.8 Ping**

4831 **D.8.1 Introduction**

4832 The resource using which an Client keeps its Connection with an Server active.

4833 **D.8.2 Wellknown URI**

4834 /oic/ping

4835 **D.8.3 Resource Type**

4836 The resource type (rt) is defined as: oic.wk.ping.

4837 **D.8.4 RAML Definition**

```

4838 #%RAML 0.8
4839 title: Ping
4840 version: v1-20160622
4841 traits:
4842   - interface :
4843     queryParameters:
4844       if:
4845         enum: ["oic.if.rw", "oic.if.baseline"]
4846
4847 /oic/ping:
4848   description: |
4849     The resource using which an Client keeps its Connection with an Server active.
4850
4851   is : ['interface']
4852   get:
4853     description: |
4854       Retrieve the ping information
4855
4856   responses :
4857     200:
4858       body:
4859         application/json:
4860           schema: /
4861             {
4862               "$schema": "http://json-schemas.org/draft-04/schema#",
4863               "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
4864 rights reserved.",
4865               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.ping-
4866 schema.json#",
4867               "definitions": {
4868                 "oic.wk.ping": {
4869                   "type": "object",
4870                   "properties": {
4871                     "in": {
4872                       "type": "integer",
4873                       "readOnly": false,
4874                       "description": "Indicates the interval for which connection shall be kept
4875 alive"

```

```

4876         }
4877     }
4878 }
4879 },
4880 "type": "object",
4881 "allOf": [
4882     { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4883     { "$ref": "#/definitions/oic.wk.ping"}
4884 ],
4885 "required": [
4886     "in"
4887 ]
4888 }
4889
4890 example: /
4891 {
4892     "rt": ["oic.wk.ping"],
4893     "n": "Ping Information",
4894     "in": 16
4895 }
4896
4897 post:
4898     description: |
4899         Update or reset the alive interval
4900
4901     body:
4902         application/json:
4903             schema: /
4904                 {
4905                     "$schema": "http://json-schemas.org/draft-04/schema#",
4906                     "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All rights
4907 reserved.",
4908                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.ping-schema.json#",
4909                     "definitions": {
4910                         "oic.wk.ping": {
4911                             "type": "object",
4912                             "properties": {
4913                                 "in": {
4914                                     "type": "integer",
4915                                     "readOnly": false,
4916                                     "description": "Indicates the interval for which connection shall be kept
4917 alive"
4918                                 }
4919                             }
4920                         },
4921                     },
4922                     "type": "object",
4923                     "allOf": [
4924                         { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4925                         { "$ref": "#/definitions/oic.wk.ping"}
4926                     ],
4927                     "required": [
4928                         "in"
4929                     ]
4930                 }
4931
4932             example: /
4933                 {
4934                     "in": 16
4935                 }
4936
4937     responses :
4938         203:
4939             description: |

```

4940 Successfully updated & restarted alive interval timer.

4941

4942 D.8.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
in	integer		Read Write	Indicates the interval for which connection shall be kept alive

4943 D.8.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/ping		get	post		

4944 D.9 Discoverable Resources, baseline interface

4945 D.9.1 Introduction

4946 Baseline representation of /oic/res; list of discoverable resources

4947 D.9.2 Wellknown URI

4948 /oic/res

4949 D.9.3 Resource Type

4950 The resource type (rt) is defined as: oic.wk.res.

4951 D.9.4 RAML Definition

4952 `##RAML 0.8`

4953 `title: Discoverable Resources`

4954 `version: v1-20160622`

4955 `traits:`

4956 `- interface-ll :`

4957 `queryParameters:`

4958 `if:`

4959 `enum: ["oic.if.ll"]`

4960 `- interface-baseline :`

4961 `queryParameters:`

4962 `if:`

4963 `enum: ["oic.if.baseline"]`

4964

4965 `/oic-res-baseline-URI:`

4966 `description: |`

4967 `Baseline representation of /oic/res; list of discoverable resources`

4968

4969 `is : ['interface-baseline']`

4970 `get:`

4971 `description: |`

4972 `Retrieve the discoverable resource set, baseline interface`

4973

4974 `responses :`

4975 `200:`

4976 `body:`

4977 `application/json:`

4978 `schema: |`

4979 `{`

4980 `"$schema": "http://json-schema.org/draft-v4/schema#",`

4981 `"description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All`

```

4982 rights reserved.",
4983 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-
4984 schema.json#",
4985 "definitions": {
4986   "oic.res-baseline": {
4987     "type": "object",
4988     "properties": {
4989       "rt": {
4990         "type": "array",
4991         "items": {
4992           "type": "string",
4993           "maxLength": 64
4994         },
4995         "minItems": 1,
4996         "readOnly": true,
4997         "description": "Resource Type"
4998       },
4999       "if": {
5000         "type": "array",
5001         "items": {
5002           "type": "string",
5003           "enum": ["oic.if.baseline", "oic.if.ll"]
5004         },
5005         "minItems": 1,
5006         "readOnly": true,
5007         "description": "The interface set supported by this resource"
5008       },
5009       "n": {
5010         "type": "string",
5011         "maxLength": 64,
5012         "readOnly": true,
5013         "description": "Human friendly name"
5014       },
5015       "mpro": {
5016         "readOnly": true,
5017         "description": "Supported messaging protocols",
5018         "type": "string",
5019         "maxLength": 64
5020       },
5021       "links": {
5022         "type": "array",
5023         "items": {
5024           "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
5025         }
5026       }
5027     },
5028     "required": ["rt", "if", "links"]
5029   }
5030 },
5031 "description": "The list of resources expressed as OIC links",
5032 "type": "array",
5033 "items": {
5034   "$ref": "#/definitions/oic.res-baseline"
5035 }
5036 }
5037
5038 example: /
5039 [
5040   {
5041     "rt": ["oic.wk.res"],
5042     "if": ["oic.if.baseline", "oic.if.ll"],
5043     "links":
5044       [
5045         {
5046           "href": "/humidity",
5047           "rt": ["oic.r.humidity"],
5048           "if": ["oic.if.s"],
5049           "p": {"bm": 3},
5050           "eps": [
5051             {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},

```

```

5052         {"ep": "coaps://[fe80::b1d6]:1122"},
5053         {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5054     ],
5055 },
5056 {
5057     "href": "/temperature",
5058     "rt": ["oic.r.temperature"],
5059     "if": ["oic.if.s"],
5060     "p": {"bm": 3},
5061     "eps": [
5062         {"ep": "coap://[2001:db8:a::123]:2222"}
5063     ]
5064 }
5065 ]
5066 }
5067 ]
5068 ]

```

5069 D.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	yes	Read Only	Resource Type
n	string		Read Only	Human friendly name
links	array: see schema	yes	Read Write	
mpro	string		Read Only	Supported messaging protocols
if	array: see schema	yes	Read Only	The interface set supported by this resource

5070 D.9.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/res		get			

5071 D.10 Discoverable Resources, link list interface

5072 D.10.1 Introduction

5073 Link list representation of /oic/res; list of discoverable resources

5074 D.10.2 Wellknown URI

5075 /oic/res

5076 D.10.3 Resource Type

5077 The resource type (rt) is defined as: oic.wk.res.

5078 D.10.4 RAML Definition

```

5079 #%RAML 0.8
5080 title: Discoverable Resources
5081 version: v1-20160622
5082 traits:
5083   - interface-ll :
5084     queryParameters:
5085       if:
5086         enum: ["oic.if.ll"]
5087   - interface-baseline :
5088     queryParameters:
5089       if:

```

```

5090         enum: ["oic.if.baseline"]
5091
5092 /oic-res-ll-URI:
5093     description: |
5094         Link list representation of /oic/res; list of discoverable resources
5095
5096     is : ['interface-ll']
5097
5098     get:
5099         description: |
5100             Retrieve the discoverable resource set, link list interface
5101
5102     responses :
5103         200:
5104             body:
5105                 application/json:
5106                     schema: /
5107                         {
5108                             "$schema": "http://json-schema.org/draft-v4/schema#",
5109                             "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
rights reserved.",
5110                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-schema-
5111 ll.json#",
5112                             "definitions": {
5113                                 "oic.wk.res-ll": {
5114                                     "description": "The list of resources expressed as OCF links without di",
5115                                     "type": "array",
5116                                     "items": {
5117                                         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
5118                                     }
5119                                 }
5120                             },
5121                             "allOf": [
5122                                 {"$ref": "#/definitions/oic.wk.res-ll"}
5123                             ]
5124                         }
5125
5126     example: /
5127     [
5128         {
5129             "href": "/humidity",
5130             "rt": ["oic.r.humidity"],
5131             "if": ["oic.if.s"],
5132             "p": {"bm": 3},
5133             "eps": [
5134                 {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
5135                 {"ep": "coaps://[fe80::b1d6]:1122"},
5136                 {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5137             ]
5138         },
5139         {
5140             "href": "/temperature",
5141             "rt": ["oic.r.temperature"],
5142             "if": ["oic.if.s"],
5143             "p": {"bm": 3},
5144             "eps": [
5145                 {"ep": "coap://[[2001:db8:a::123]:2222"}
5146             ]
5147         }
5148     ]
5149
5150

```

D.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
---------------	------------	-----------	-------------	-------------

rt	array: see schema	yes	Read Write	Resource Type
di	multiple types: see schema		Read Write	Unique identifier for device (UUID)
title	string		Read Write	A title for the link relation. Can be used by the UI to provide a context
eps	array: see schema	yes	Read Write	the Endpoint information of the target Resource
ins	multiple types: see schema		Read Write	The instance identifier for this web link in an array of web links - used in collections
p	object: see schema		Read Write	Specifies the framework policies on the Resource referenced by the target URI
href	string	yes	Read Write	This is the target URI, it can be specified as a Relative Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique.
rel	multiple types: see schema		Read Write	The relation of the target URI referenced by the link to the context URI
type	array: see schema		Read Write	A hint at the representation of the resource referenced by the target URI. This represents the media types that are used for both accepting and emitting
anchor	string		Read Write	This is used to override the context URI e.g. override the URI

				of the containing collection
if	array: schema	see	yes	Read Write The interface set supported by this resource

5151 **D.10.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/res		get			

5152 **D.10.7 Referenced JSON schemas**

5153 **D.10.8 oic.oic-link-schema.json**

```

5154 {
5155   "$schema": "http://json-schema.org/draft-04/schema#",
5156   "description": "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All rights
5157 reserved.",
5158   "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
5159   "definitions": {
5160     "oic.oic-link": {
5161       "type": "object",
5162       "properties": {
5163         "href": {
5164           "type": "string",
5165           "maxLength": 256,
5166           "description": "This is the target URI, it can be specified as a Relative Reference or
5167 fully-qualified URI. Relative Reference should be used along with the di parameter to make it
5168 unique.",
5169           "format": "uri"
5170         },
5171         "rel": {
5172           "oneOf": [
5173             {
5174               "type": "array",
5175               "items": {
5176                 "type": "string",
5177                 "maxLength": 64
5178               },
5179               "minItems": 1,
5180               "default": ["hosts"]
5181             },
5182             {
5183               "type": "string",
5184               "maxLength": 64,
5185               "default": "hosts"
5186             }
5187           ],
5188           "description": "The relation of the target URI referenced by the link to the context URI"
5189         },
5190         "rt": {
5191           "type": "array",
5192           "items": {
5193             "type": "string",
5194             "maxLength": 64
5195           },
5196           "minItems": 1,
5197           "description": "Resource Type"
5198         },
5199         "if": {
5200           "type": "array",
5201           "items": {
5202             "type": "string",
5203             "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
5204 "oic.if.a", "oic.if.s" ]
5205           },
5206           "minItems": 1,
5207           "description": "The interface set supported by this resource"
5208         },

```

```

5209     "di": {
5210         "$ref": "oic.types-schema.json#/definitions/uuid",
5211         "description": "Unique identifier for device (UUID)"
5212     },
5213     "p": {
5214         "description": "Specifies the framework policies on the Resource referenced by the target
5215 URI",
5216         "type": "object",
5217         "properties": {
5218             "bm": {
5219                 "description": "Specifies the framework policies on the Resource referenced by the
5220 target URI for e.g. observable and discoverable",
5221                 "type": "integer"
5222             },
5223         },
5224         "required" : ["bm"]
5225     },
5226     "title": {
5227         "type": "string",
5228         "maxLength": 64,
5229         "description": "A title for the link relation. Can be used by the UI to provide a
5230 context"
5231     },
5232     "anchor": {
5233         "type": "string",
5234         "maxLength": 256,
5235         "description": "This is used to override the context URI e.g. override the URI of the
5236 containing collection",
5237         "format": "uri"
5238     },
5239     "ins": {
5240         "oneOf": [
5241             {
5242                 "type": "integer",
5243                 "description": "An ordinal number that is not repeated - must be unique in the
5244 collection context"
5245             },
5246             {
5247                 "type": "string",
5248                 "maxLength": 256,
5249                 "format": "uri",
5250                 "description": "Any unique string including a URI"
5251             },
5252             {
5253                 "$ref": "oic.types-schema.json#/definitions/uuid",
5254                 "description": "Unique identifier (UUID)"
5255             }
5256         ],
5257         "description": "The instance identifier for this web link in an array of web links - used
5258 in collections"
5259     },
5260     "type": {
5261         "type": "array",
5262         "description": "A hint at the representation of the resource referenced by the target
5263 URI. This represents the media types that are used for both accepting and emitting",
5264         "items" : {
5265             "type": "string",
5266             "maxLength": 64
5267         },
5268         "minItems": 1,
5269         "default": "application/cbor"
5270     },
5271     "eps": {
5272         "type": "array",
5273         "description": "the Endpoint information of the target Resource",
5274         "items": {
5275             "type": "object",
5276             "properties": {
5277                 "ep": {
5278                     "type": "string",
5279                     "format": "uri",

```

```

5280         "description": "URI with Transport Protocol Suites + Endpoint Locator as specified
5281 in 10.2.1"
5282     },
5283     "pri": {
5284         "type": "integer",
5285         "minimum": 1,
5286         "description": "The priority among multiple Endpoints as specified in 10.2.3"
5287     }
5288 }
5289 }
5290 }
5291 },
5292 "required": [ "href", "rt", "if", "eps" ]
5293 }
5294 },
5295 "type": "object",
5296 "allOf": [
5297     { "$ref": "#/definitions/oic.oic-link" }
5298 ]
5299 }
5300

```

5301 **D.11 Scenes (Top level)**

5302 **D.11.1 Introduction**

5303 Toplevel Scene resource. This resource is a generic collection resource. The rts value shall contain
5304 oic.wk.scenecollection resource types.

5305 **D.11.2 Example URI**

5306 /SceneListResURI

5307 **D.11.3 Resource Type**

5308 The resource type (rt) is defined as: oic.wk.scenelist.

5309 **D.11.4 RAML Definition**

```

5310 #%RAML 0.8
5311 title: Scene
5312 version: v1-20160622
5313 traits:
5314   - interface :
5315       queryParameters:
5316           if:
5317               enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
5318
5319 /SceneListResURI:
5320     description: |
5321       Toplevel Scene resource.
5322       This resource is a generic collection resource.
5323       The rts value shall contain oic.wk.scenecollection resource types.
5324
5325     get:
5326       description: |
5327         Provides the current list of web links pointing to scenes
5328
5329     responses :
5330       200:
5331         body:
5332           application/json:
5333             schema: /

```

```

5334     {
5335         "$schema": "http://json-schema.org/draft-04/schema#",
5336         "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5337 reserved.",
5338         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
5339 schema.json#",
5340         "title": "Collection",
5341         "definitions": {
5342             "oic.collection.setoflinks": {
5343                 "description": "A set (array) of simple or individual OIC Links. In
5344 addition to properties required for an OIC Link, the identifier for that link in this set is also
5345 required",
5346                 "type": "array",
5347                 "items": {
5348                     "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
5349                 }
5350             },
5351             "oic.collection.alllinks": {
5352                 "description": "All forms of links in a collection",
5353                 "oneOf": [
5354                     {
5355                         "$ref": "#/definitions/oic.collection.setoflinks"
5356                     }
5357                 ]
5358             },
5359             "oic.collection": {
5360                 "type": "object",
5361                 "description": "A collection is a set (array) of tagged-link or set
5362 (array) of simple links along with additional properties to describe the collection itself",
5363                 "properties": {
5364                     "id": {
5365                         "anyOf": [
5366                             {
5367                                 "type": "integer",
5368                                 "description": "A number that is unique to that
5369 collection; like an ordinal number that is not repeated"
5370                             },
5371                             {
5372                                 "type": "string",
5373                                 "description": "A unique string that could be a hash or
5374 similarly unique"
5375                             },
5376                             {
5377                                 "$ref": "oic.types-schema.json#/definitions/uuid",
5378                                 "description": "A unique string that could be a UUIDv4"
5379                             }
5380                         ],
5381                         "description": "ID for the collection. Can be a value that is
5382 unique to the use context or a UUIDv4"
5383                     },
5384                     "di": {
5385                         "$ref": "oic.types-schema.json#/definitions/uuid",
5386                         "description": "The device ID which is an UUIDv4 string; used for
5387 backward compatibility with Spec A definition of /oic/res"
5388                     },
5389                     "rts": {
5390                         "$ref": "oic.core-
5391 schema.json#/definitions/oic.core/properties/rt",
5392                         "description": "Defines the list of allowable resource types (for
5393 Target and anchors) in links included in the collection; new links being created can only be from
5394 this list"
5395                     },
5396                     "drel": {
5397                         "type": "string",
5398                         "description": "When specified this is the default relationship
5399 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
5400                     },
5401                     "links": {
5402                         "$ref": "#/definitions/oic.collection.alllinks"
5403                     }
5404                 }
5405             }
5406         }
5407     }

```

```

5405     },
5406     "type": "object",
5407     "allof": [
5408         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
5409         {"$ref": "#/definitions/oic.collection"}
5410     ]
5411 }
5412
5413 example: /
5414 {
5415     "rt": ["oic.wk.scenelist"],
5416     "n": "list of scene Collections",
5417     "rts": ["oic.wk.scenecollection"],
5418     "links": [
5419     ]
5420 }
5421

```

5422 D.11.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
drel	string		Read Write	When specified this is the default relationship to use when an OIC Link does not specify an explicit relationship with *rel* parameter
links	multiple types: see schema		Read Write	
id	multiple types: see schema		Read Write	ID for the collection. Can be an value that is unique to the use context or a UUIDv4
rts	multiple types: see schema		Read Write	Defines the list of allowable resource types (for Target and anchors) in links included in the collection; new links being created can only be from this list
di	multiple types: see schema		Read Write	The device ID which is an UUIDv4 string; used for backward compatibility with Spec A definition of /oic/res

5423 D.11.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
----------	--------	------	--------	--------	--------

/SceneListResURI		get			
------------------	--	-----	--	--	--

5424 D.12 Scene Collections

5425 D.12.1 Introduction

5426 Collection that models a set of Scenes. This resource is a generic collection resource with
5427 additional parameters. The rts value shall contain oic.sceneMember resource types. The additional
5428 parameters are lastScene, this is the scene value last set by any OIC Client sceneValueList,
5429 this is the list of available scenes lastScene shall be listed in sceneValueList.

5430 D.12.2 Example URI

5431 /SceneCollectionResURI

5432 D.12.3 Resource Type

5433 The resource type (rt) is defined as: oic.wk.scenecollection.

5434 D.12.4 RAML Definition

```

5435 #%RAML 0.8
5436 title: Scene
5437 version: v1-20160622
5438 traits:
5439   - interface :
5440     queryParameters:
5441       if:
5442         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
5443
5444 /SceneCollectionResURI:
5445   description: |
5446     Collection that models a set of Scenes.
5447     This resource is a generic collection resource with additional parameters.
5448     The rts value shall contain oic.sceneMember resource types.
5449     The additional parameters are
5450     lastScene, this is the scene value last set by any OIC Client
5451     sceneValueList, this is the list of available scenes
5452     lastScene shall be listed in sceneValueList.
5453
5454   get:
5455     description: |
5456       Provides the current list of web links pointing to scenes
5457
5458     responses :
5459       200:
5460         body:
5461           application/json:
5462             schema: /
5463               {
5464                 "$schema": "http://json-schema.org/draft-04/schema#",
5465                 "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
5466 rights reserved.",
5467                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
5468 schema.json#",
5469                 "title" : "Scene Collection",
5470                 "definitions": {
5471                   "oic.sceneCollection": {
5472                     "type": "object",
5473                     "properties": {
5474                       "lastScene": {
5475                         "type": "string",
5476                         "description": "Last selected Scene, shall be part of sceneValues"

```

```

5477         },
5478         "sceneValues": {
5479             "type": "string",
5480             "readOnly": true,
5481             "description": "All available scene values"
5482         },
5483         "n": {
5484             "type": "string",
5485             "description": "Used to name the Scene collection"
5486         },
5487         "id": {
5488             "type": "string",
5489             "description": "A unique string that could be a hash or
5490 similarly unique"
5491         },
5492         "rts": {
5493             "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
5494             "description": "Defines the list of allowable resource types in links
5495 included in the collection; new links being created can only be from this list"
5496         },
5497         "links": {
5498             "type": "array",
5499             "description": "Array of OIC web links that are reference from this
5500 collection",
5501             "items": {
5502                 "allOf": [
5503                     { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
5504                     { "required": [ "ins" ] }
5505                 ]
5506             }
5507         },
5508     },
5509     "required": [ "lastScene", "sceneValues", "rts", "id" ]
5510 }
5511 },
5512 "type": "object",
5513 "allOf" : [
5514     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5515     { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
5516     { "$ref": "oic.oic-scene-schema.json#/definitions/oic.oic-scene-collection" }
5517 ]
5518 }
5519
5520 example: /
5521 {
5522     "lastScene": "off",
5523     "sceneValues": "off,Reading,TVWatching",
5524     "rt": ["oic.wk.scenecollection"],
5525     "n": "My Scenes for my living room",
5526     "id": "0685B960-736F-46F7-BEC0-9E6CBD671ADC1",
5527     "rts": ["oic.wk.scenemember"],
5528     "links": [
5529     ]
5530 }
5531
5532 put:
5533 description: |
5534 Provides the action to change the last settted scene selection.
5535 Calling this method shall update of all sceneMembers to the prescribed membervalue.
5536 When this method is called with the same value as the current lastScene value
5537 then all sceneMembers shall be updated.
5538
5539 body:
5540 application/json:
5541 schema: /
5542 {
5543     "$schema": "http://json-schema.org/draft-04/schema#",

```

```

5544         "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All rights
5545 reserved.",
5546         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
5547 schema.json#",
5548         "title" : "Scene Collection",
5549         "definitions": {
5550             "oic.sceneCollection": {
5551                 "type": "object",
5552                 "properties": {
5553                     "lastScene": {
5554                         "type": "string",
5555                         "description": "Last selected Scene, shall be part of sceneValues"
5556                     },
5557                     "sceneValues": {
5558                         "type": "string",
5559                         "readOnly": true,
5560                         "description": "All available scene values"
5561                     },
5562                     "n": {
5563                         "type": "string",
5564                         "description": "Used to name the Scene collection"
5565                     },
5566                     "id": {
5567                         "type": "string",
5568                         "description" : "A unique string that could be a hash or
5569 similarly unique"
5570                     },
5571                     "rts": {
5572                         "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
5573                         "description": "Defines the list of allowable resource types in links included
5574 in the collection; new links being created can only be from this list"
5575                     },
5576                     "links": {
5577                         "type": "array",
5578                         "description": "Array of OIC web links that are reference from this
5579 collection",
5580                         "items" : {
5581                             "allOf": [
5582                                 { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
5583                                 { "required" : [ "ins" ] }
5584                             ]
5585                         }
5586                     },
5587                     "required": [ "lastScene" ]
5588                 }
5589             },
5590             "type": "object",
5591             "allOf" : [
5592                 { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5593                 { "$ref": "#/definitions/oic.sceneCollection" }
5594             ]
5595         }
5596     }
5597 }
5598
5599     example: /
5600     {
5601         "lastScene": "Reading"
5602     }
5603
5604     responses :
5605     200:
5606         description: |
5607             Indicates that the value is changed.
5608             The changed properties are provided in the response.
5609
5610         body:
5611             application/json:

```

```

5612     schema: /
5613     {
5614         "$schema": "http://json-schema.org/draft-04/schema#",
5615         "description": "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
5616 rights reserved.",
5617         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
5618 schema.json#",
5619         "title": "Scene Collection",
5620         "definitions": {
5621             "oic.sceneCollection": {
5622                 "type": "object",
5623                 "properties": {
5624                     "lastScene": {
5625                         "type": "string",
5626                         "description": "Last selected Scene, shall be part of sceneValues"
5627                     },
5628                     "sceneValues": {
5629                         "type": "string",
5630                         "readOnly": true,
5631                         "description": "All available scene values"
5632                     },
5633                     "n": {
5634                         "type": "string",
5635                         "description": "Used to name the Scene collection"
5636                     },
5637                     "id": {
5638                         "type": "string",
5639                         "description": "A unique string that could be a hash or
5640 similarly unique"
5641                     },
5642                     "rts": {
5643                         "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
5644                         "description": "Defines the list of allowable resource types in links
5645 included in the collection; new links being created can only be from this list"
5646                     },
5647                     "links": {
5648                         "type": "array",
5649                         "description": "Array of OIC web links that are reference from this
5650 collection",
5651                     "items": {
5652                         "allOf": [
5653                             { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
5654                             { "required": [ "ins" ] }
5655                         ]
5656                     }
5657                 },
5658                 "required": [ "lastScene" ]
5659             }
5660         },
5661         "type": "object",
5662         "allOf": [
5663             { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5664             { "$ref": "#/definitions/oic.sceneCollection" }
5665         ]
5666     }
5667 }
5668 }
5669
5670     example: /
5671     {
5672         "lastScene": "Reading"
5673     }
5674

```

5675 D.12.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
lastScene	string	yes	Read Write	Last selected Scene, shall be

				part of sceneValues
links	array: see schema		Read Write	Array of OIC web links that are reference from this collection
sceneValues	string	yes	Read Only	All available scene values
n	string		Read Write	Used to name the Scene collection
rts	multiple types: see schema	yes	Read Write	Defines the list of allowable resource types in links included in the collection; new links being created can only be from this list
id	string	yes	Read Write	A unique string that could be a hash or similarly unique

5676 **D.12.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/SceneCollectionResURI	put	get			

5677 **D.13 Scene Member**

5678 **D.13.1 Introduction**

5679 Collection that models a scene member.

5680 **D.13.2 Example URI**

5681 /SceneMemberResURI

5682 **D.13.3 Resource Type**

5683 The resource type (rt) is defined as: oic.wk.scenemember.

5684 **D.13.4 RAML Definition**

```

5685 #%RAML 0.8
5686 title: Scene
5687 version: v1-20160622
5688 traits:
5689   - interface :
5690     queryParameters:
5691       if:
5692         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
5693
5694 /SceneMemberResURI:
5695   description: |
5696     Collection that models a scene member.
5697
5698   get:
5699     description: |

```

```

5700         Provides the scene member
5701
5702     responses :
5703         200:
5704             body:
5705                 application/json:
5706                     schema: /
5707                         {
5708                             "$schema": "http://json-schema.org/draft-04/schema#",
5709                             "description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All
rights reserved.",
5710                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneMember-
schema.json#",
5711                             "title" : "Scene Member",
5712                             "definitions": {
5713                                 "oic.sceneMember": {
5714                                     "type": "object",
5715                                     "properties": {
5716                                         "n": {
5717                                             "type": "string",
5718                                             "description": "Used to name the Scene collection"
5719                                         },
5720                                         "id": {
5721                                             "type": "string",
5722                                             "description": "Can be an value that is unique to the use context or a
UUIDv4"
5723                                         },
5724                                         "SceneMappings" : {
5725                                             "type": "array",
5726                                             "description": "array of mappings per scene, can be 1",
5727                                             "items": {
5728                                                 "type": "object",
5729                                                 "properties": {
5730                                                     "scene": {
5731                                                         "type": "string",
5732                                                         "description": "Specifies a scene value that will acted upon"
5733                                                     },
5734                                                     "memberProperty": {
5735                                                         "type": "string",
5736                                                         "readOnly": true,
5737                                                         "description": "property name that will be mapped"
5738                                                     },
5739                                                     "memberValue": {
5740                                                         "type": "string",
5741                                                         "readOnly": true,
5742                                                         "description": "value of the Member Property"
5743                                                     }
5744                                                 }
5745                                             },
5746                                             "required": [ "scene", "memberProperty", "memberValue" ]
5747                                         },
5748                                         "link": {
5749                                             "type": "string",
5750                                             "description": "web link that points at a resource",
5751                                             "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
5752                                         }
5753                                     },
5754                                     "required": [ "link" ]
5755                                 }
5756                             },
5757                             "type": "object",
5758                             "allOf" : [
5759                                 { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5760                                 { "$ref": "#/definitions/oic.sceneMember" }
5761                             ]
5762                         }

```

```

5767     }
5768
5769     example: /
5770     {
5771       "rt": ["oic.wk.sceneMember"],
5772       "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
5773       "n": "my binary switch (for light bulb) mappings",
5774       "link": {
5775         "href": "binarySwitch",
5776         "rt": ["oic.r.switch.binary"],
5777         "if": ["oic.if.a", "oic.if.baseline"],
5778         "eps": [
5779           {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
5780           {"ep": "coaps://[fe80::b1d6]:1122"},
5781           {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5782         ]
5783       },
5784       "sceneMappings": [
5785         {
5786           "scene": "off",
5787           "memberProperty": "value",
5788           "memberValue": true
5789         },
5790         {
5791           "scene": "Reading",
5792           "memberProperty": "value",
5793           "memberValue": false
5794         },
5795         {
5796           "scene": "TVWatching",
5797           "memberProperty": "value",
5798           "memberValue": true
5799         }
5800       ]
5801     }
5802

```

5803 D.13.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
SceneMappings	array: see schema		Read Write	array of mappings per scene, can be 1
link	string	yes	Read Write	web link that points at a resource
id	string		Read Write	Can be an value that is unique to the use context or a UUIDv4
n	string		Read Write	Used to name the Scene collection

5804 D.13.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/SceneMemberResURI		get			

5805 D.14 Resource directory resource

5806 D.14.1 Introduction

5807 Resource to be exposed by any Device that can act as a Resource Directory

5808 **D.14.2 Wellknown URI**

5809 /oic/rd

5810 **D.14.3 Resource Type**

5811 The resource type (rt) is defined as: oic.wk.rd.

5812 **D.14.4 RAML Definition**

5813 `##RAML 0.8`

5814 `title: Resource Directory`

5815 `version: v1-20160622`

5816 `traits:`

5817 `- rddefinterface :`

5818 `queryParameters:`

5819 `if:`

5820 `enum: ["oic.if.baseline"]`

5821 `description: Interface is optional since there is only one interface supported for the`

5822 `Resource Type`

5823 `Both for RD selectin and for publish`

5824 `default: oic.if.baseline`

5825

5826

5827 `/oic/rd:`

5828 `description: |`

5829 `Resource to be exposed by any Device that can act as a Resource Directory`

5830

5831 `get:`

5832 `description: |`

5833 `Get the attributes of the Resource Directory for selection purposes.`

5834

5835 `queryParameters:`

5836 `rt:`

5837 `enum: oic.wk.rd`

5838 `type: string`

5839 `description: Only one Resource Type is used for GET; RT is optional`

5840

5841 `required: false`

5842 `example: GET /oic/rd?rt=oic.wk.rd`

5843

5844 `is : ['rddefinterface']`

5845 `responses :`

5846 `200:`

5847 `description: |`

5848 `Respond with the selector criteria - either the set of attributes or the bias factor`

5849

5850 `body:`

5851 `application/json:`

5852 `schema: /`

5853 `{`

5854 `"$schema": "http://json-schema.org/draft-04/schema#",`

5855 `"description" : "Copyright (c) 2016, 2017 Open Connectivity Foundation, Inc. All`

5856 `rights reserved.",`

5857 `"id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.selection-`

5858 `schema.json#",`

5859 `"title" : "RD Selection",`

5860 `"definitions": {`

5861 `"oic.rd.attributes": {`

5862 `"type": "object",`

5863 `"properties": {`

```

5864         "n": {
5865             "type": "string",
5866             "description": "A human friendly name for the Resource Directory"
5867         },
5868         "di": {
5869             "$ref": "oic.types-schema.json#/definitions/uuid",
5870             "description": "A unique identifier for the Resource Directory - the same
5871 as the device ID of the RD"
5872         },
5873     },
5874     "sel": {
5875         "description": "Selection criteria that a device wanting to publish to any
5876 RD can use to choose this Resource Directory over others that are discovered",
5877         "oneOf": [
5878             {
5879                 "type": "object",
5880                 "properties": {
5881                     "pwr": {
5882                         "type": "string",
5883                         "enum": [ "ac", "batt", "safe" ],
5884                         "description": "A hint about how the RD is powered. If AC then this
5885 is stronger than battery powered. If source is reliable (safe) then appropriate mechanism for
5886 managing power failure exists"
5887                     },
5888                     "conn": {
5889                         "type": "string",
5890                         "enum": [ "wrld", "wrlds" ],
5891                         "description": "A hint about the networking connectivity of the RD.
5892 *wrld* if wired connected and *wrlds* if wireless connected."
5893                     },
5894                     "bw": {
5895                         "type": "string",
5896                         "description": "Qualitative bandwidth of the connection",
5897                         "enum": [ "high", "low", "lossy" ]
5898                     },
5899                     "mf": {
5900                         "type": "integer",
5901                         "description": "Memory factor - Ratio of available memory to total
5902 memory expressed as a percentage"
5903                     },
5904                     "load": {
5905                         "type": "array",
5906                         "items": {
5907                             "type": "number"
5908                         },
5909                         "minitems": 3,
5910                         "maxitems": 3,
5911                         "description": "Current load capacity of the RD. Expressed as a
5912 load factor 3-tuple (upto two decimal points each). Load factor is based on request processed in a
5913 1 minute, 5 minute window and 15 minute window"
5914                     }
5915                 }
5916             },
5917             {
5918                 "type": "integer",
5919                 "minimum": 0,
5920                 "maximum": 100,
5921                 "description": "A bias factor calculated by the Resource directory -
5922 the value is in the range of 0 to 100 - 0 implies that RD is not to be selected. Client chooses RD
5923 with highest bias factor or randomly between RDs that have same bias factor"
5924             }
5925         ]
5926     }
5927 }
5928 }
5929 },
5930 "type": "object",
5931 "allOf": [ {"$ref": "#/definitions/oic.rd.attributes"} ],
5932 "required": ["sel"]
5933 }
5934

```

```

5935         example: /
5936             {
5937                 "rt": "oic.wk.rd",
5938                 "sel": 50
5939             }
5940
5941     post:
5942         description: |
5943             Publish the resource information
5944             Appropriates parts of the information posted will be discovered through /oic/res
5945
5946     queryParameters:
5947         rt:
5948             enum: oic.wk.rdpub
5949             type: string
5950
5951             description: Only one Resource Type is used for GET; RT is optional
5952
5953             required: false
5954
5955             example: GET /oic/rd?rt=oic.wk.rdpub
5956
5957     is : ['rddefinterface']
5958     body:
5959         application/json:
5960             schema: /
5961                 {
5962                     "$schema": "http://json-schema.org/draft-04/schema#",
5963                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5964 reserved.",
5965                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
5966 schema.json#",
5967                     "title": "RD Publish & Update",
5968                     "definitions": {
5969                         "oic.rd.publish": {
5970                             "description": "Publishes resources as OIC Links into the resource directory",
5971                             "properties": {
5972                                 "linkSet": {
5973                                     "$ref": "oic.collection-schema.json#/definitions/oic.collection.setoflinks"
5974                                 },
5975                                 "ttl": {
5976                                     "type": "integer",
5977                                     "description": "Time to indicate a RD, how long to keep this published item.
5978 After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
5979 publishing device updates the ttl using the update schema"
5980                                 }
5981                             },
5982                             "type": "object",
5983                             "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
5984                             "required": [ "links" ],
5985                             "dependencies": {
5986                                 "links": [ "ttl" ]
5987                             }
5988                         }
5989                     }
5990
5991     responses :
5992         200:
5993             description: |
5994                 Respond with the same schema as publish but with the links have the "ins" parameter set
5995 to the appropriate instance value.
5996                 This value is used by the receiver to manage that OIC Link instance.

```

```

5997     body:
5998         application/json:
5999             schema: /
6000                 {
6001                     "$schema": "http://json-schema.org/draft-04/schema#",
6002                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
6003 reserved.",
6004                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
6005 schema.json#",
6006                     "title": "RD Publish & Update",
6007                     "definitions": {
6008                         "oic.rd.publish": {
6009                             "description": "Publishes resources as OIC Links into the resource directory",
6010                             "properties": {
6011                                 "linkSet": {
6012                                     "$ref": "oic.collection-schema.json#/definitions/oic.collection.setoflinks"
6013                                 },
6014                                 "ttl": {
6015                                     "type": "integer",
6016                                     "description": "Time to indicate a RD, how long to keep this published
6017 item. After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
6018 publishing device updates the ttl using the update schema"
6019                                 }
6020                             }
6021                         },
6022                     },
6023                     "type": "object",
6024                     "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
6025                     "required": [ "links" ],
6026                     "dependencies": {
6027                         "links": [ "ttl" ]
6028                     }
6029                 }
6030
6031             example: /
6032                 {
6033                     "links": [
6034                         {
6035                             "href": "coap://someAuthority:1000/somePath",
6036                             "rt": "oic.r.someResource",
6037                             "if": "oic.if.a",
6038                             "ins": 12345
6039                         },
6040                         {
6041                             "href": "coap://someAuthority:1000/somePath",
6042                             "rt": "oic.r.someOtherResource",
6043                             "if": "oic.if.baseline",
6044                             "ins": 54321
6045                         }
6046                     ],
6047                     "ttl": 600
6048                 }
6049
6050     delete:
6051         description: |
6052             Delete a particular OIC Link - the link may be a simple link or a link in a tagged set.
6053
6054     queryParameters:
6055         di:
6056             type: string
6057
6058             description: This is used to determine which set of links to operata on. (Need
6059 authentication to ensure that there is no spoofing). If instance is ommitted then the entire set of
6060 links from this device ID is deleted
6061
6062         required: true

```

```

6062         example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"
6063
6064         ins:
6065             type: string
6066             description: Instance of the link to delete
6067         Value of parameter is a string where instance to be deleted are comma separated
6068
6069         required: false
6070         example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1";ins="20"
6071
6072         responses :
6073             200:
6074                 description: |
6075                     The delete succeeded
6076
6077

```

D.14.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
di	multiple types: see schema		Read Write	A unique identifier for the Resource Directory - the same as the device ID of the RD
sel	multiple types: see schema	yes	Read Write	Selection criteria that a device wanting to publish to any RD can use to choose this Resource Directory over others that are discovered
n	string		Read Write	A human friendly name for the Resource Directory

D.14.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/rd		get	post	delete	

D.15 Icon

D.15.1 Introduction

This resource describes the attributes associated with an Icon.

D.15.2 Example URI

/IconResURI

D.15.3 Resource Type

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

6086 D.15.4 RAML Definition

```
6087 #%RAML 0.8
6088 title: OICIcon
6089 version: v1.1.0-20161107
6090 traits:
6091   - interface :
6092     queryParameters:
6093       if:
6094         enum: ["oic.if.r", "oic.if.baseline"]
6095
6096 /IconResURI:
6097   description: |
6098     This resource describes the attributes associated with an Icon.
6099
6100   is : ['interface']
6101   get:
6102     description: |
6103       Retrieves the current icon properties.
6104
6105   responses :
6106     200:
6107       body:
6108         application/json:
6109           schema: /
6110             {
6111               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.r.icon.json#",
6112               "$schema": "http://json-schema.org/draft-04/schema#",
6113               "description" : "Copyright (c) 2016,2017 Open Connectivity Foundation, Inc. All
6114 rights reserved.",
6115               "title": "Icon",
6116               "definitions": {
6117                 "oic.r.icon": {
6118                   "properties": {
6119                     "mimetype": {
6120                       "type": "string",
6121                       "maxLength": 64,
6122                       "readOnly": true,
6123                       "description": "Specifies the format of the MIME Type"
6124                     },
6125                     "width": {
6126                       "type": "integer",
6127                       "minimum": 1,
6128                       "readOnly": true,
6129                       "description": "Specifies the width in pixels"
6130                     },
6131                     "height": {
6132                       "type": "integer",
6133                       "minimum": 1,
6134                       "readOnly": true,
6135                       "description": "Specifies the height in pixels"
6136                     },
6137                     "media": {
6138                       "type": "string",
6139                       "maxLength": 256,
6140                       "format" : "uri",
6141                       "readOnly": true,
6142                       "description": "Specifies the media URL to icon"
6143                     }
6144                   }
6145                 }
6146             },
6147           "type": "object",
```

```

6148     "allOf": [
6149       { "$ref": "oic.core-schema.json#/definitions/oic.core" },
6150       { "$ref": "#/definitions/oic.r.icon" }
6151     ],
6152     "required": ["mimetype", "width", "height", "media"]
6153   }
6154
6155   example: /
6156     {
6157       "rt": ["oic.r.icon"],
6158       "id": "unique_example_id",
6159       "mimetype": "image/png",
6160       "width": 256,
6161       "height": 256,
6162       "media": "http://findbetter.ru/public/uploads/1481662800/2043.png"
6163     }
6164

```

6165 D.15.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
mimetype	string	yes	Read Only	Specifies the format of the MIME Type
width	integer	yes	Read Only	Specifies the width in pixels
media	string	yes	Read Only	Specifies the media URL to icon
height	integer	yes	Read Only	Specifies the height in pixels

6166 D.15.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/IconResURI		get			

6167 D.16 Introspection Resource

6168 D.16.1 Introduction

6169 This resource provides the means to get the device introspection data specifying all the endpoints
6170 of the device. The url hosted by this resource is either a local or an external url.

6171 D.16.2 Example URI

6172 /IntrospectionResURI

6173 D.16.3 Resource Type

6174 The resource type (rt) is defined as: oic.wk.introspection.

6175 D.16.4 RAML Definition

```

6176 #%RAML 0.8
6177 title: OICIntrospection
6178 version: v1.0.0-20160707
6179 traits:
6180   - interface :
6181     queryParameters:
6182       if:
6183         enum: ["oic.if.baseline"]
6184
6185 /IntrospectionResURI:

```

```

6186     description: |
6187         This resource provides the means to get the device introspection data specifying all the
6188         endpoints of the device.
6189         The url hosted by this resource is either a local or an external url.
6190
6191     is : ['interface']
6192
6193     get:
6194         responses :
6195             200:
6196                 body:
6197                     application/json:
6198                         schema: /
6199                             {
6200                                 "id": "http://www.openconnectivity.org/ocf-
apis/core/schemas/oic.wk.introspectionInfo.json#",
6201                                 "$schema": "http://json-schema.org/draft-04/schema#",
6202                                 "description" : "Copyright (c) 2017 Open Interconnect Consortium, Inc. All rights
6203                                 reserved.",
6204                                 "title": "introspection resource",
6205                                 "definitions": {
6206                                     "oic.wk.introspectionInfo": {
6207                                         "type": "object",
6208                                         "properties": {
6209                                             "urlInfo": {
6210                                                 "type": "array",
6211                                                 "description": "The valid range for the value Property",
6212                                                 "readOnly": true,
6213                                                 "minItems": 1,
6214                                                 "items": {
6215                                                     "type" : "object",
6216                                                     "properties": {
6217                                                         "url": {
6218                                                             "type": "string",
6219                                                             "format": "uri",
6220                                                             "description" : "url to download the description"
6221                                                         },
6222                                                         "protocol": {
6223                                                             "type": "string",
6224                                                             "enum": [ "coap", "coaps", "http", "https", "coap+tcp",
6225                                                             "coaps+tcp" ],
6226                                                             "description" : "protocol to be used to download the introspection"
6227                                                         },
6228                                                         "content-type": {
6229                                                             "type": "string",
6230                                                             "enum": [ "application/json", "application/cbor" ],
6231                                                             "default" : "application/cbor",
6232                                                             "description" : "content-type of the introspection data"
6233                                                         },
6234                                                         "version": {
6235                                                             "type": "integer",
6236                                                             "enum": [ 1 ],
6237                                                             "default" : 1,
6238                                                             "description" : "version the introspection data that can be
6239                                                             downloaded"
6240                                                         }
6241                                                     },
6242                                                     "required" : [ "url","protocol"]
6243                                                 }
6244                                             },
6245                                             "required" : ["urlInfo"]
6246                                         },
6247                                         "type": "object",
6248                                         "allOf": [
6249                                             {"$ref": "#/definitions/oic.wk.introspectionInfo"},
6250                                             {"$ref": "oic.core-schema.json#/definitions/oic.core"}
6251                                         ]
6252                                 }
6253

```

```

6254     }
6255
6256     example: /
6257     {
6258         "rt" : ["oic.wk.introspection"],
6259         "urlInfo" : [
6260             {
6261                 "content-type" : "application/cbor",
6262                 "protocol" : "coap",
6263                 "url" : "/IntrospectionExampleURI"
6264             }
6265         ]
6266     }
6267

```

6268 **D.16.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
urlInfo	array: schema see	yes	Read Only	The valid range for the value Property

6269 **D.16.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/IntrospectionResURI		get			

6270

