



OIC CORE SPECIFICATION V1.0.0 Part 1

Open Interconnect Consortium (OIC)
admin@openinterconnect.org

Legal Disclaimer

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

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

Copyright © 2015 Open Interconnect Consortium, Inc. All rights reserved.

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

CONTENTS

25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

1	Scope	12
2	Normative references	12
3	Terms, definitions, symbols and abbreviations	15
3.1	Terms and definitions	15
3.2	Symbols and abbreviations	17
3.3	Conventions	18
3.4	Data types	19
4	Document conventions and organization	19
5	OIC architecture	20
5.1	Overview	20
5.2	Principle	21
5.3	OIC functional block diagram	23
5.3.1	OIC Framework	24
5.4	Example Scenario with OIC Roles	24
5.5	Example Scenario: Bridging to Non-OIC ecosystem	25
6	Identification and addressing	26
6.1	Introduction	26
6.2	Identification	27
6.2.1	Resource identification and addressing	27
6.3	Namespace:	28
6.4	Network addressing	28
7	Resource model	29
7.1	OIC resource model	29
7.1.1	Introduction	29
7.1.2	OIC Resource	29
7.1.3	OIC Interface	30
7.1.4	OIC Resource Properties	30
7.1.5	Resource representation	32
7.1.6	Structure	33
7.2	Usage of OIC resource model	43
7.2.1	Values for common properties	43
7.2.2	OIC Core Resources	47
8	CRUDN	48
8.1	Overview	48
8.2	CREATE	48
8.2.1	CREATE request	49
8.2.2	Processing by the OIC Server	49
8.2.3	CREATE response	49
8.3	RETRIEVE	50
8.3.1	RETRIEVE request	50
8.3.2	Processing by the OIC Server	50

68	8.3.3	RETRIEVE response	50
69	8.4	UPDATE	51
70	8.4.1	UPDATE request	51
71	8.4.2	Processing by the OIC Server	51
72	8.4.3	UPDATE response	51
73	8.5	DELETE	51
74	8.5.1	DELETE request	52
75	8.5.2	Processing by the OIC Server	52
76	8.5.3	DELETE response	52
77	8.6	NOTIFY	52
78	9	Network and connectivity	53
79	9.1	Introduction	53
80	9.2	Architecture	53
81	9.3	IPv6 network layer requirements	54
82	9.3.1	Introduction	54
83	9.3.2	IPv6 node requirements	55
84	9.3.3	IPv6 router	56
85	9.3.4	IPv6 host	56
86	9.3.5	IPv6 constrained nodes	56
87	10	Endpoint discovery	56
88	10.1	Introduction	56
89	10.2	CoAP based Endpoint discovery	56
90	11	Functional interactions	58
91	11.1	Introduction	58
92	11.2	Provisioning	59
93	11.3	Resource discovery	62
94	11.3.1	Introduction	62
95	11.3.2	Resource based discovery: mechanisms	62
96	11.3.3	Resource based discovery: Information publication process	64
97	11.3.4	Resource based discovery: Finding information	65
98	11.3.5	Resource discovery using '/oic/res'	72
99	11.3.6	Resource directory (RD) based discovery	73
100	11.4	Notification	82
101	11.4.1	Overview	82
102	11.4.2	Observe	82
103	11.5	Device management	84
104	11.5.1	Monitoring	84
105	11.5.2	Diagnostics and maintenance	86
106	11.5.3	Security considerations for device management	88
107	11.6	Scenes, Rules and Scripts	88
108	11.6.1	Introduction	88
109	11.6.2	Scenes	89
110	11.6.3	Rules	93
111	11.6.4	Security considerations	98

112	12	Messaging.....	98
113	12.1	Introduction	98
114	12.2	Mapping of CRUDN to CoAP	98
115	12.2.1	Overview.....	98
116	12.2.2	Request methods	98
117	12.2.3	Content Type negotiation	99
118	12.2.4	CRUDN to CoAP response codes.....	100
119	12.2.5	CoAP block transfer	100
120	12.2.6	CoAP serialization over TCP	101
121	12.3	Mapping of CRUDN to HTTP	102
122	12.3.1	Supported HTTP features.....	102
123	12.3.2	Supported HTTP methods	102
124	12.3.3	Supported HTTP header fields	102
125	12.3.4	Content Type negotiation	105
126	12.3.5	HTTP response codes.....	106
127	12.3.6	Method mapping.....	106
128	13	Security.....	106
129	14	Multi resource model support	107
130	14.1	Interoperability issue	107
131	14.1.1	Multiple IoT Standards	107
132	14.1.2	Different resource models	107
133	14.2	A scheme to exchange resource model information	109
134	14.2.1	A scheme to exchange resource model information.....	109
135	14.2.2	New Content-Formats (Internet Media Type) for OIC resource model.....	109
136	Annex A (informative) Operation Examples.....		110
137	A.1	Introduction	110
138	A.2	When at home: From smartphone turn on a single light	110
139	A.3	GroupAction execution	111
140	A.4	When garage door opens, turn on lights in hall; also notify smartphone.....	111
141	A.5	Device management.....	111
142	Annex B (informative) OIC interaction scenarios and deployment models.....		113
143	B.1	OIC interaction scenarios	113
144	B.2	OIC deployment model	114
145	Annex C (informative) Other Resource Models and OIC Mapping		116
146	C.1	Multiple resource models.....	116
147	C.2	OIC approach for support of multiple resource models.....	116
148	C.3	Resource model indication.....	117
149	C.4	An Example Profile (IPSO profile).....	117
150	C.4.1	Conceptual equivalence	117
151	Annex D (informative) Resource type definitions.....		120
152	D.1	List of resource type definitions	120
153	D.2	OIC Configuration.....	120
154	D.2.1	Introduction	120

155	D.2.2	Wellknown URI.....	120
156	D.2.3	Resource Type.....	120
157	D.2.4	RAML Definition.....	120
158	D.2.5	Property Definition.....	123
159	D.2.6	CRUDN behavior.....	123
160	D.3	OIC Logical Device.....	123
161	D.3.1	Introduction.....	123
162	D.3.2	Wellknown URI.....	123
163	D.3.3	Resource Type.....	123
164	D.3.4	RAML Definition.....	123
165	D.3.5	Property Definition.....	124
166	D.3.6	CRUDN behavior.....	124
167	D.4	OIC Inteface Types.....	124
168	D.4.1	Introduction.....	124
169	D.4.2	Wellknown URI.....	124
170	D.4.3	Resource Type.....	124
171	D.4.4	RAML Definition.....	125
172	D.4.5	Property Definition.....	125
173	D.4.6	CRUDN behavior.....	125
174	D.5	OIC Maintenance.....	126
175	D.5.1	Introduction.....	126
176	D.5.2	Wellknown URI.....	126
177	D.5.3	Resource Type.....	126
178	D.5.4	RAML Definition.....	126
179	D.5.5	Property Definition.....	128
180	D.5.6	CRUDN behavior.....	129
181	D.6	OIC Monitoring.....	129
182	D.6.1	Introduction.....	129
183	D.6.2	Wellknown URI.....	129
184	D.6.3	Resource Type.....	129
185	D.6.4	RAML Definition.....	129
186	D.6.5	Property Definition.....	130
187	D.6.6	CRUDN behavior.....	130
188	D.7	OIC Base Platform.....	130
189	D.7.1	Introduction.....	130
190	D.7.2	Wellknown URI.....	130
191	D.7.3	Resource Type.....	130
192	D.7.4	RAML Definition.....	130
193	D.7.5	Property Definition.....	132
194	D.7.6	CRUDN behavior.....	132
195	D.8	OIC Ping.....	132
196	D.8.1	Introduction.....	132
197	D.8.2	Wellknown URI.....	132
198	D.8.3	Resource Type.....	133

199	D.8.4	RAML Definition	133
200	D.8.5	Property Definition	134
201	D.8.6	CRUDN behavior	134
202	D.9	OIC Discoverable Resources	134
203	D.9.1	Introduction	134
204	D.9.2	Wellknown URI	134
205	D.9.3	Resource Type	134
206	D.9.4	RAML Definition	134
207	D.9.5	Property Definition	135
208	D.9.6	CRUDN behavior	135
209	D.10	OIC Resource Types	136
210	D.10.1	Introduction	136
211	D.10.2	Wellknown URI	136
212	D.10.3	Resource Type	136
213	D.10.4	RAML Definition	136
214	D.10.5	Property Definition	137
215	D.10.6	CRUDN behavior	137
216	D.11	Scenes (Top level)	137
217	D.11.1	Introduction	137
218	D.11.2	Wellknown URI	137
219	D.11.3	Resource Type	137
220	D.11.4	RAML Definition	137
221	D.11.5	Property Definition	141
222	D.11.6	CRUDN behavior	141
223	D.12	Scene Collections	142
224	D.12.1	Introduction	142
225	D.12.2	Wellknown URI	142
226	D.12.3	Resource Type	142
227	D.12.4	RAML Definition	142
228	D.12.5	Property Definition	149
229	D.12.6	CRUDN behavior	149
230	D.13	Scene Member	149
231	D.13.1	Introduction	149
232	D.13.2	Wellknown URI	149
233	D.13.3	Resource Type	149
234	D.13.4	RAML Definition	149
235	D.13.5	Property Definition	151
236	D.13.6	CRUDN behavior	151
237	D.14	Rules (Top level)	152
238	D.14.1	Introduction	152
239	D.14.2	Wellknown URI	152
240	D.14.3	Resource Type	152
241	D.14.4	RAML Definition	152
242	D.14.5	Property Definition	156

243	D.14.6	CRUDN behavior	156
244	D.15	Rule	156
245	D.15.1	Introduction	156
246	D.15.2	Wellknown URI.....	157
247	D.15.3	Resource Type	157
248	D.15.4	RAML Definition	157
249	D.15.5	Property Definition	163
250	D.15.6	CRUDN behavior	164
251	D.16	Rule Member	164
252	D.16.1	Introduction	164
253	D.16.2	Wellknown URI.....	164
254	D.16.3	Resource Type	164
255	D.16.4	RAML Definition	164
256	D.16.5	Property Definition	165
257	D.16.6	CRUDN behavior	165
258			
259			

260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299

Figures

Figure 1: OIC architecture - concepts	22
Figure 2: OIC functional block diagram	23
Figure 3: OIC communication layering model.....	24
Figure 4: Example illustrating the OIC Roles	25
Figure 5: OIC Framework - Architecture Detail	25
Figure 6: OIC Server bridging to Non-OIC device	26
Figure 7: JSON Schema for OIC Links.....	34
Figure 8: Example of use of anchor in OIC Link	35
Figure 9: JSON Schema for /oic/res	35
Figure 10: Example /oic/res representation.....	36
Figure 12: Example OIC Collection with simple links (JSON)	38
Figure 13: Example OIC Collection with tagged OIC Links (JSON)	38
Figure 14. Bootstrap collections	42
Figure 15. Collection as a factory	43
Figure 16. CREATE operation	49
Figure 17. RETRIEVE operation	50
Figure 18. UPDATE operation	51
Figure 19. DELETE operation	52
Figure 20. High Level Network & Connectivity Architecture.....	54
Figure 21. Provisioning State Changes.....	59
Figure 22. Interactions initiated by the OIC Device to retrieve its configuration from a configuration source	60
Figure 23. Interactions for retrieving the configuration state of an OIC Device.....	61
Figure 24. Update of and OIC Device configuration	61
Figure 25. Resource based discovery: Information publication process.....	65
Figure 26. Resource based discovery: Finding information	65
Figure 27. Indirect discovery of resource by resource directory	74
Figure 28. RD discovery and RD supported query of resources support.....	75
Figure 29. Resource Direction Deployment Scenarios	76
Figure 30. Information in a response to a query to /oic/rd	78
Figure 31. Publish – push resource information	81
Figure 32. Observe Mechanism	83
Figure 33. Retrieving all the monitoring information in a single request.....	85
Figure 34. Retrieving specific Monitoring information in multiple requests.....	86
Figure 35. Factory_Reset command	88
Figure 36 Generic scene resource structure	89
Figure 37 Interactions to check Scene support and setup of specific scenes	90

300	Figure 38 Client interactions on a specific scene	91
301	Figure 39 Interaction overview due to a Scene change	92
302	Figure 40 Clean up of Scene resource structure	92
303	Figure 41 Generic rule resource structure.....	93
304	Figure 42 Interactions to check Rule support and setup of specific rules	94
305	Figure 43 Client interactions on rules	95
306	Figure 44 Interaction overview due to a rule condition evaluating to true	96
307	Figure 45 Clean-up of rule resource structure.....	97
308	Figure 46. When at home: from smartphone turn on a single light.....	111
309	Figure 47. Device management (monitoring and maintenance)	112
310	Figure 48. Direct interaction between OIC Server and OIC Client	113
311	Figure 49. Interaction between OIC Client and OIC Server using another OIC Server.....	113
312	Figure 50. Interaction between OIC Client and OIC Server using OIC Intermediary	113
313	Figure 51. Interaction between OIC Client and OIC Server using support from multiple	
314	OIC Servers and OIC Intermediary	114
315	Figure 52. Example of OIC Devices	114

316
317
318
319
320

Tables

321		
322		
323	Table 1. Data type definition	19
324	Table 2. Example foobar Resource Type	30
325	Table 3. Example foobar properties	30
326	Table 4: Common properties for OIC Collections (in addition to Common Properties	
327	defined in section 7.2.1)	39
328	Table 5: OIC-defined Interfaces for OIC Collection	40
329	Table 6. Resource type property definition.....	44
330	Table 7. Resource interface property definition.....	44
331	Table 8. OIC standard interface.....	45
332	Table 9. Policy Property Definition.....	47
333	Table 10. Name Property Definition	47
334	Table 11. Parameters of CRUDN messages	48
335	Table 12. List of OIC Core Resources	58
336	Table 13. Configuration Resources.....	62
337	Table 14. oic.wk.con resource type definition	62
338	Table 15. Mandatory discovery OIC Core Resources.....	66
339	Table 16. oic.wk.res resource type definition	67

340	Table 17. Protocol scheme registry.....	67
341	Table 18. oic.wk.d resource type definition	68
342	Table 19. oic.wk.p resource type definition	69
343	Table 20. Optional discovery OIC Core Resources	70
344	Table 21. oic.wk.rts supported resource type definition.....	71
345	Table 22. oic.wk.ifs resource type definition	71
346	Table 23. oic.wk.ad resource type definition	71
347	Table 25. Optional monitoring device management OIC Core Resources.....	84
348	Table 26. oic.wk.mon resource type definition	85
349	Table 27. Optional diagnostics and maintenance device management OIC Core	
350	Resources	86
351	Table 28. oic.wk.mnt resource type definition	86
352	Table 29 list of resource types for Scenes	93
353	Table 30 List of resource types part of Rules.....	97
354	Table 31. CoAP request methods	98
355	Table 32. Content Types and Content Formats	100
356	Table 33. Ping resource	102
357	Table 34. oic.wk.ping resource type definition	102
358	Table 35. HTTP header fields usage in OIC	103
359	Table 36. HTTP response codes.....	106
360	Table 37. oic.example.light resource type definition.....	110
361	Table 38. oic.ex.garagedoor resource type definition.....	110
362	Table 39. Light control resource type definition.....	118
363	Table 40. Light control resource type definition.....	118
364		
365		
366		
367		

368 1 Scope

369 The OIC specifications are divided into two sets of documents:

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

378 This document is the OIC Core specification which specifies the OIC Framework and core
379 architecture.

380

381 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

434 IETF RFC 7230, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, June
435 2014.
436 <https://tools.ietf.org/html/rfc7230.txt>

437 IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, June 2014.
438 <https://tools.ietf.org/html/rfc7231.txt>

439 IETF RFC 7232, *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*, June 2014.
440 <https://tools.ietf.org/html/rfc7232.txt>

441 IETF RFC 7233, *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*, June 2014.
442 <https://tools.ietf.org/html/rfc7233.txt>

443 IETF RFC 7234, *Hypertext Transfer Protocol (HTTP/1.1): Caching*, June 2014.
444 <https://tools.ietf.org/html/rfc7234.txt>

445 IETF RFC 7235, *Hypertext Transfer Protocol (HTTP/1.1): Authentication*, June 2014.
446 <https://tools.ietf.org/html/rfc7235.txt>

447 IETF RFC 7252, *The Constrained Application Protocol (CoAP)*, June 2014
448 <http://tools.ietf.org/html/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 draft-ietf-6lo-btle-14, *IPv6 over BLUETOOTH(R) Low Energy*, June 25, 2015
455 <http://www.ietf.org/id/draft-ietf-6lo-btle-14.txt>

456 IETF draft-ietf-core-resource-directory-02, *CoRE Resource Directory*, November 9, 2014
457 <http://www.ietf.org/id/draft-ietf-core-resource-directory-02.txt>

458 IETF draft-ietf-core-observe-16, *Observing Resources in CoAP*, December 30, 2014
459 <http://www.ietf.org/id/draft-ietf-core-observe-16.txt>

460 IETF draft-ietf-core-block-18, *Block-wise transfers in CoAP*, September 14, 2015
461 <http://www.ietf.org/id/draft-ietf-core-block-18.txt>

462 IETF draft-ietf-core-interfaces-02, *CoRE Interfaces*, November 9, 2014
463 <http://www.ietf.org/id/draft-ietf-core-interfaces-02.txt>

464 IETF draft-tschofenig-core-coap-tcp-tls-04, *A TCP and TLS Transport for the Constrained*
465 *Application Protocol (CoAP)*, June 10 2015
466 <https://www.ietf.org/id/draft-tschofenig-core-coap-tcp-tls-04.txt>

467 IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00, *Auto-Configuration of a Network of Hybrid*
468 *Unicast/Multicast DNS-Based Service Discovery Proxy Nodes*, March 5 2015
469 <https://tools.ietf.org/html/draft-ietf-homenet-hybrid-proxy-zeroconf-00>

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

472 OIC Security, *Open Interconnect Consortium Security Capabilities*, Version 1.0,

473 UPnP AV CDS, *UPnP AV Content Directory Service, Version 4*
474 <http://upnp.org/specs/av/UPnP-av-ContentDirectory-v4-Service.pdf>

475

476 **3 Terms, definitions, symbols and abbreviations**

477 **3.1 Terms and definitions**

478 **3.1.1**

479 **OIC Core Resources**

480 those OIC Resources that are defined in this specification

481 **3.1.2**

482 **Configuration Source**

483 an entity in the Cloud or Service Network or a local read-only file which contains and provides

484 configuration related information to the OIC Devices

485 **3.1.3**

486 **Entity**

487 an element of the physical world that is exposed through an OIC Device

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

489 **3.1.4**

490 **Observe**

491 the act of monitoring an OIC Resource by sending a RETRIEVE request which is cached by the

492 OIC Server hosting the OIC Resource and reprocessed on every change to that OIC Resource

493 **3.1.5**

494 **OIC Client**

495 a logical entity that accesses an OIC Resource on an OIC Server

496 **3.1.6**

497 **OIC Collection**

498 an OIC Resource that contains zero or more OIC Links

499 **3.1.7**

500 **OIC Device**

501 a logical entity that assumes one or more OIC roles (OIC Client, OIC Server)

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

503 **3.1.8**

504 **OIC Functionality**

505 the base/core functionality contained in any OIC Device

506 **3.1.9**

507 **OIC Framework**

508 a set of common functionalities and interactions defined in this specification, which enable

509 interoperability across a wide range of networked devices, including IoT

510 **3.1.10**

511 **OIC Infrastructure Gateway**

512 an OIC Platform that ensures interoperability between OIC Devices by including the following:

513 – Resource Directory

514 **3.1.11**

515 **OIC Platform**

516 a physical device containing one or more OIC Devices

517 **3.1.12**
518 **OIC Links**
519 extends typed web links as specified in IETF RFC 5988

520 **3.1.13**
521 **OIC Resource**
522 represents an artifact modelled and exposed by the OIC Framework

523 **3.1.14**
524 **OIC Resource Interface**
525 a qualification of the permitted requests on an OIC Resource

526 **3.1.15**
527 **OIC Resource Property**
528 a significant aspect or notion including metadata that is exposed through the OIC Resource

529 **3.1.16**
530 **OIC Resource Type**
531 a uniquely named definition of class of OIC Resource Properties and the interactions that is
532 supported by that class

533 Note 1 to entry: Each OIC Resource has an OIC Property "rt" whose value is the unique name of the OIC Resource
534 Type.

535 **3.1.17**
536 **OIC Server**
537 a logical entity with the role of providing resource state information and facilitating remote
538 interaction with its resources

539 Note 1 to entry: An OIC Server can be implemented to expose non-OIC Device resources to OIC Clients (section 5.5)

540 **3.1.18**
541 **Non OIC Device**
542 A device which does not comply to the OIC specifications

543 **3.1.19**
544 **Notification**
545 the mechanism to make an OIC Client aware of resource state changes in an OIC Resource

546 **3.1.20**
547 **RAE Client**
548 an OIC Client which supports XMPP functionality

549 **3.1.21**
550 **Remote Access Endpoint (RAE) Server**
551 an OIC Server which supports XMPP and it can publish its resource(s) to an XMPP server, thus
552 becoming remotely addressable and accessible

553 Note 1 to entry: It also supports ICE/STUN/TURN if the application on the OIC Server requires it.

554 **3.1.22**
555 **Resource Directory**
556 an OIC Device that hosts descriptions of resources held on other OIC Servers, allowing lookups
557 to be performed for those resources

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

560 **3.1.23**
561 **Scene**
562 a single value listed in Scene Values

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

565 **3.1.24**

566 **Scene Collection**

567 an OIC Resource that contains an enumeration of possible Scene Values and the current Scene
568 Value

569 Note 1 to entry: This resource is a collection resource with additional data, and the member values of the scene resource
570 are scene members.

571 **3.1.25**

572 **Scene Value**

573 a Scene enumerator representing the state in which an OIC Resource can be

574 **3.1.26**

575 **Scene Member**

576 an OIC Resource that contains mappings of Scene Values to values of a property in the resource

577 **3.1.27**

578 **Rule**

579 an OIC Resource that contains a condition which when evaluated as true will launch a Script in an
580 OIC Server

581 **3.1.28**

582 **Rule Condition**

583 an expression describing how to evaluate a resource property against a value

584 Note 1 to entry: The Rule Condition is expressed in EBNF and uses references to a property in a resource on a specified
585 OIC Server.

586 **3.1.29**

587 **Rule Member**

588 an OIC resource that contains the values of a property in the resource that are set when an
589 associated Rule Condition is true.

590 **3.1.30**

591 **Script**

592 a set of Rule Members to be executed when a Rule Condition holds true

593 **3.1.31**

594 **Default Interface**

595 The first interface listed within the interface ('if') property for a specific resource in /oic/res is the
596 Default Interface. If the resource is not exposed in /oic/res then the Default Interface is the interface
597 as defined for that resource in an OIC Specification. When an interface is omitted in a request, the
598 Default Interface will be the interface used to generate the response.

599 **3.2 Symbols and abbreviations**

600 **3.2.1**

601 **BLE**

602 Bluetooth Low Energy

603 **3.2.2**

604 **CBOR**

605 Concise Binary Object Representation

606 **3.2.3**

607 **CoAP**

608 Constrained Application Protocol

609 **3.2.4**
610 **EXI**
611 Efficient XML Interchange

612 **3.2.5**
613 **IRI**
614 Internationalized Resource Identifiers

615 **3.2.6**
616 **ISP**
617 Internet Service Provider

618 **3.2.7**
619 **JSON**
620 JavaScript Object Notation

621 **3.2.8**
622 **mDNS**
623 Multicast Domain Name Service

624 **3.2.9**
625 **MTU**
626 Maximum Transmission Unit

627 **3.2.10**
628 **NAT**
629 Network Address Translation

630 **3.2.11**
631 **OIC**
632 Open Interconnect Consortium

633 the organization that created this specification

634 **3.2.12**
635 **URI**
636 Uniform Resource Identifier

637 **3.2.13**
638 **URN**
639 Uniform Resource Name

640 **3.2.14**
641 **UTC**
642 Coordinated Universal Time

643 **3.2.15**
644 **UUID**
645 Universal Unique Identifier

646 **3.2.16**
647 **XML**
648 Extensible Markup Language

649 **3.3 Conventions**
650 In this specification a number of terms, conditions, mechanisms, sequences, parameters, events,
651 states, or similar terms are printed with the first letter of each word in uppercase and the rest

652 lowercase (e.g., Network Architecture). Any lowercase uses of these words have the normal
 653 technical English meaning.

654 3.4 Data types

655 Table 1 contains the definitions of data types used to describe an OIC Resource. The data types
 656 are derived from JSON values as defined in ECMA-4-4. However an OIC resource can overload a
 657 JSON defined value to specify a particular subset of the JSON value. These OIC specific data
 658 types are defined in Table 1. The OIC data types can be adapted for a particular usage, for example
 659 the length of a string can be changed for a specific usage.

660 **Table 1. Data type definition**

Name	JSON value	Description
boolean	false true	Binary-value {0, 1}.
BSV	string	A blank (i.e. space) separated list of values encoded within a string. The value type in the BSV is described by the property where the BSV is used. For example a BSV of integers.
CSV	string	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.
date	string	As defined in ISO 8601. The format is [yyyy]-[mm]-[dd].
datetime	string	As defined in ISO 8601. Concatenation of “date” and “time” with the “T” as a delimiter between “date” and “time”. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.
enum	enum	Enumerated type.
float	number	Signed IEEE 754 single precision float value.
integer	number	Signed 32 bit integer.
json	object/array	A data represented using a JSON element which could be an object or array as defined in ECMA-4-4. The JSON object or array needs to be described by means of a JSON schema.
string	string	Character string shall not exceed a max length of 64 octets (bytes).
time	string	As defined in ISO 8601 but restricted to UTC with a trailing “Z”. The format is [hh]:[mm]:[ss]Z.
URI	string	A uniform resource identifier (URI) is a string of characters used to identify a resource. The URI value shall not exceed a max length of 256 octets (bytes).
UUID	string	An identifier formatted according to IETF RFC 4122.

661

662 4 Document conventions and organization

663 In this document, features are described as required, recommended, allowed or DEPRECATED as
 664 follows:

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

- 666 • These basic features shall be implemented to comply with OIC Core Architecture. The phrases
 667 “shall not”, and “PROHIBITED” indicate behavior that is prohibited, i.e. that if performed means
 668 the implementation is not in compliance.

669 Recommended (or should)(S).

- 670 • These features add functionality supported by OIC Core Architecture and should be
 671 implemented. Recommended features take advantage of the capabilities OIC Core Architecture,

672 usually without imposing major increase of complexity. Notice that for compliance testing, if a
673 recommended feature is implemented, it shall meet the specified requirements to be in
674 compliance with these guidelines. Some recommended features could become requirements
675 in the future. The phrase “should not” indicates behavior that is permitted but not recommended.

676 Allowed (may or allowed)(O).

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

680 DEPRECATED.

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

687 Conditionally allowed (CA)

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

690 Conditionally required (CR)

- 691 • The definition or behaviour depends on a condition. If the specified condition is met, then the
692 definition or behaviour is required. Otherwise the definition or behaviour is allowed as default
693 unless specifically defined as not allowed.

694

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

696 Words that are emphasized are printed in italic.

697 **5 OIC architecture**

698 **5.1 Overview**

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

704 Specifically, the OIC architecture provides:

- 705 • A communication and interoperability framework for multiple market segments (Consumer,
706 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,
707 transports and use cases
- 708 • A common and consistent model for describing the environment and enabling information
709 and semantic interoperability
- 710 • Common communication protocols for discovery and connectivity
- 711 • Common security and identification mechanisms

- 712 • Opportunity for innovation and product differentiation
- 713 • A scalable solution addressing different device capabilities, applicable to smart devices as
- 714 well as the smallest connected things and wearable devices

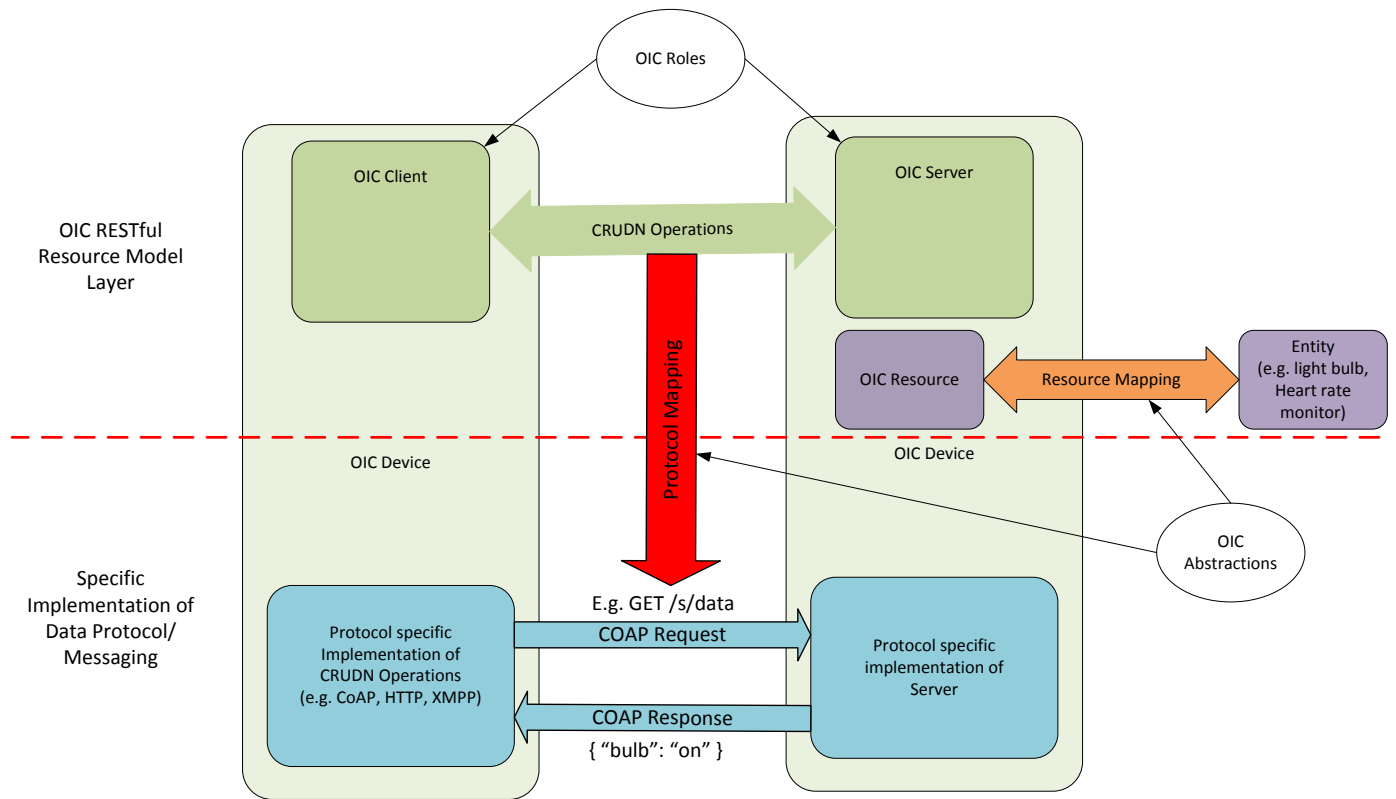
715 The OIC architecture is based on the Resource Oriented Architecture design principles and
716 described in the sections 5.2 through 5.5 respectively. Section 5.2 presents the guiding principles
717 for OIC operations. Section 5.3 defines the OIC functional block diagram and OIC Framework.
718 Section 5.4 provides an example scenario with OIC Roles. Section 5.5 provides an example
719 scenario of bridging to non-OIC ecosystem.

720 **5.2 Principle**

721 In the OIC architecture, Entities in the physical world (e.g., temperature sensor, an electric light or
722 a home appliance) are represented as resources. Interactions with an Entity are achieved through
723 its resource representations (section 7.1.5) using operations that adhere to Representational State
724 Transfer (REST) architectural style, i.e., RESTful interactions.

725 The OIC architecture defines the overall structure of the OIC Framework as an information system
726 and the interrelationships of the Entities that make up OIC. Entities are exposed as OIC Resources,
727 with their unique identifiers (URIs) and support interfaces that enable RESTful operations on the
728 OIC Resources. Every RESTful operation has an initiator of the operation (the client) and a
729 responder to the operation (the server). In the OIC Framework, the notion of the client and server
730 is realized through OIC Roles (section 5.4). Any OIC Device can act as an OIC Client and initiate
731 a RESTful operation on any OIC Device acting as an OIC Server. Likewise, any OIC Device that
732 exposes Entities as OIC Resources acts as an OIC Server. Conformant to the REST architectural
733 style, each RESTful operation in OIC contains all the information necessary to understand the
734 context of the interaction and is driven using a small set of generic operations, i.e., Create, Read,
735 Update, Delete, Notify (CRUDN) defined in section 8, which include representations of OIC
736 Resources.

737 Figure 1 depicts the OIC architecture.



738
739

740

741

Figure 1: OIC architecture - concepts

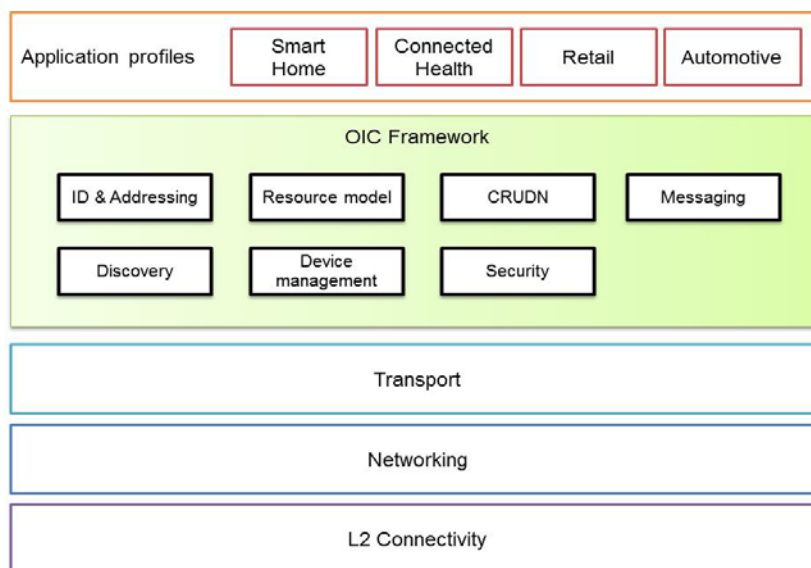
742 The architecture is organized conceptually into three major aspects that provide overall separation
743 of concern: resource model, RESTful operations and abstractions.

- 744 • Resource model: The resource model provides the abstractions and concepts required to
745 logically model, and logically operate on the application and its environment. The resource
746 model is agnostic to an application domain like smart home, industrial or automotive. For
747 example, the resource model defines an OIC Resource which abstracts an Entity and the
748 representation of an OIC Resource maps the Entity's state. Other resource model concepts
749 can be used to model other aspects, for example behavior.
- 750 • RESTful operations: The generic CRUDN operations are defined using the RESTful paradigm
751 to model the interactions with an OIC Resource in a protocol and technology agnostic way. The
752 specific communication or messaging protocols are part of the protocol abstraction and
753 mapping of OIC Resources to specific protocols is provided in section 12.
- 754 • Abstraction: The abstractions in the resource model and the RESTful operations are mapped
755 to concrete elements using abstraction primitives. An entity handler is used to map an Entity
756 to an OIC Resource and connectivity abstraction primitives are used to map logical RESTful
757 operations to data connectivity protocols or technologies. Entity handlers may also be used
758 to map OIC Resources to Entities that are reached over protocols that are not natively supported
759 by OIC.

760

761 **5.3 OIC functional block diagram**

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



765

766

Figure 2: OIC functional block diagram

767 • **L2 connectivity:** Provides the functionalities required for establishing physical and data
768 link layer connections (e.g., Wi-Fi™ or Bluetooth™ connection) to the network.

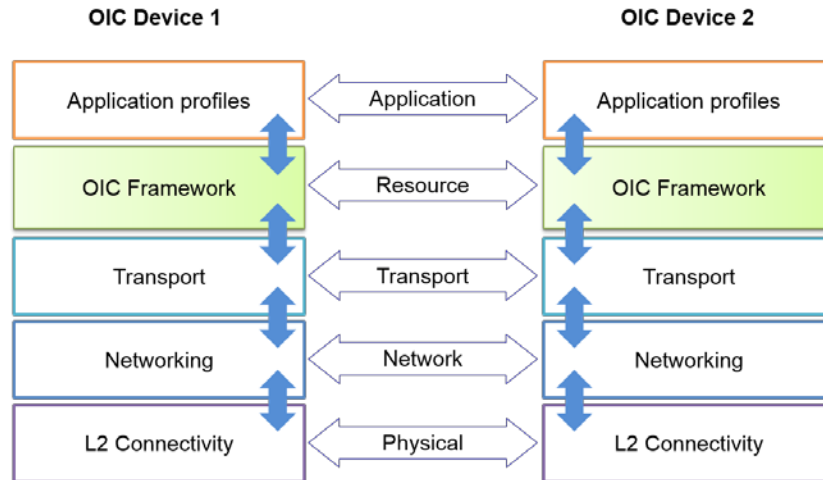
769 • **Networking:** Provides functionalities required for OIC Devices to exchange data among
770 themselves over the network (e.g., Internet).

771 • **Transport:** Provides end-to-end flow transport with specific QoS constraints. Examples of
772 a transport protocol include TCP and UDP or new Transport protocols under development
773 in the IETF, e.g., Delay Tolerant Networking (DTN).

774 • **OIC Framework:** Provides the OIC core functionalities as defined in this specification. The
775 functional block is the source of requests and responses that are the content of the
776 communication between two OIC Devices.

777 • **Application profile:** Provides market segment specific data model and functionalities, e.g.,
778 smart home data model and functions for the smart home market segment.

779 When two OIC Devices communicate with each other, each functional block in an OIC Device
780 interacts with its counterpart in the peer OIC Device as shown in Figure 3.



781
782 **Figure 3: OIC communication layering model**

783 **5.3.1 OIC Framework**

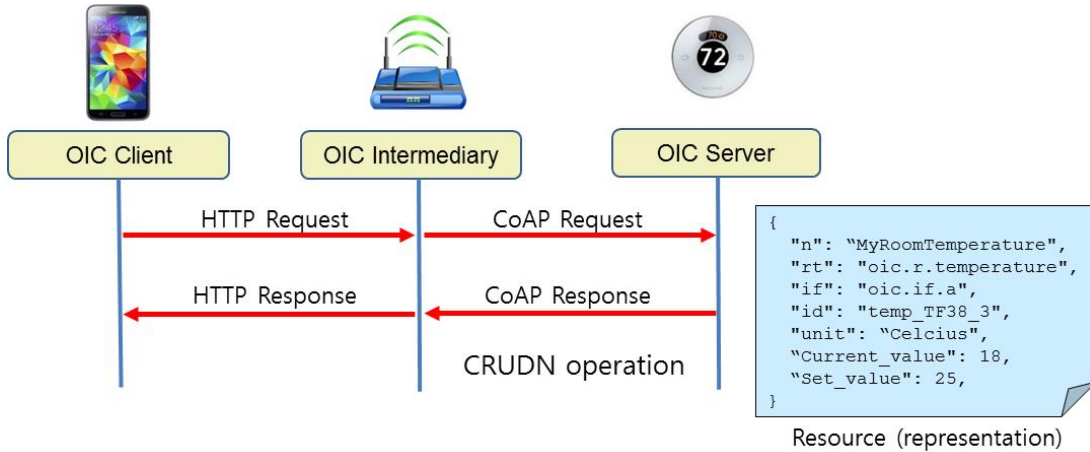
784 OIC Framework consists of functions which provide core functionalities for OIC operation.

- 785 1) **Identification and addressing.** Defines the identifier and addressing capability. The
786 Identification and addressing function is defined in section 6.
- 787 2) **Discovery.** Defines the process for discovering available
788 a) OIC Devices (Endpoint Discovery in section 10) and
789 b) OIC Resources (Resource Discovery in section 11.3)
- 790 3) **Resource model.** Specifies the capability for representation of Entities in terms of resources
791 and defines mechanisms for manipulating the resources. The resource model function is
792 defined in section 7.
- 793 4) **CRUDN.** Provides a generic scheme for the interactions between an OIC Client and OIC Server
794 as defined in section 8.
- 795 5) **Messaging.** Provides specific message protocols for RESTful operation, i.e. CRUDN. For
796 example, CoAP is a primary messaging protocol. The messaging function is defined in section
797 12.
- 798 6) **Device management.** Specifies the discipline of managing the capabilities of an OIC Device,
799 and includes device provisioning and initial setup as well as device monitoring and diagnostics.
800 The device management function is defined in section 11.5.
- 801 7) **Security.** Includes authentication, authorization, and access control mechanisms required for
802 secure access to Entities. The security function is defined in section 13.

803 **5.4 Example Scenario with OIC Roles**

804 OIC interactions are defined between logical entities known as OIC Roles. Three roles are defined:
805 OIC Client, OIC Server and OIC Intermediary.

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



811

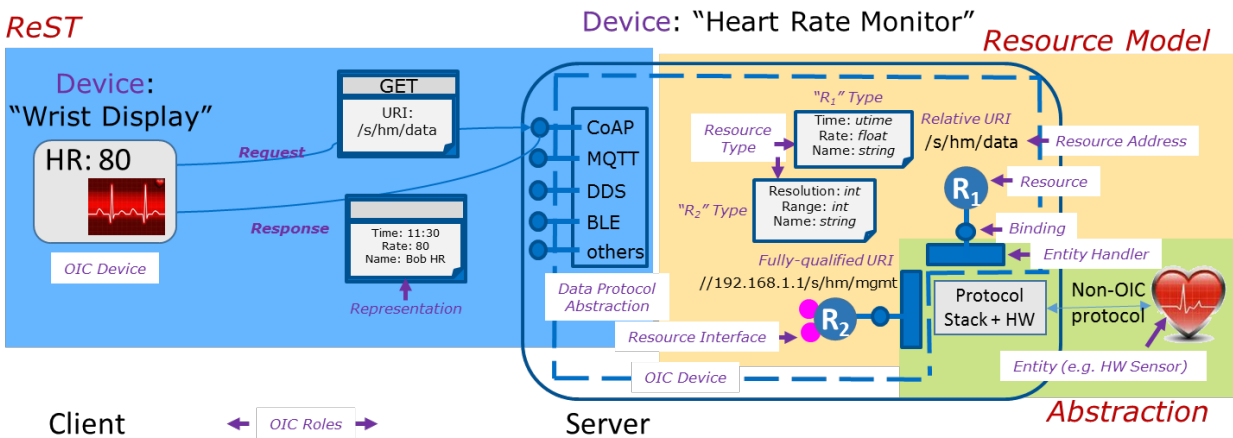
812

Figure 4: Example illustrating the OIC Roles

813 **5.5 Example Scenario: Bridging to Non-OIC ecosystem**

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

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



817

818

Figure 5: OIC Framework - Architecture Detail

819

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



822
823

824

Figure 6: OIC Server bridging to Non-OIC device

825 On start-up the OIC Server runs the entity handlers which discover the non-OIC systems (e.g.,
826 Heart Rate Sensor Device) and create resources for each device or functionality discovered. The
827 entity handler creates an OIC Resource for each discovered device or functionality and binds itself
828 to that OIC Resource. These resources are made discoverable by the OIC Server.

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

834 **6 Identification and addressing**

835 **6.1 Introduction**

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

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

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

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

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

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

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

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

859 6.2 Identification

860 An identifier shall be unique within the context or domain of use. There are many schemes that
861 may be used to generate an identifier that has the required properties. The identifier may be
862 context-specific in that the identifier is expected to be and guaranteed to be unique only within that
863 context or domain. Identifier may also be context-independent where these identifiers are
864 guaranteed to be unique across all contexts and domains both spatially and temporally. The
865 context-specific identifiers could be defined by simple schemes like monotonic enumeration or may
866 be defined by overloading an address or name, for example an IP address may be an identifier
867 within the private domain behind a gateway in a smart home. On the other hand, context-
868 independent identifiers require a stronger scheme that derives universally unique identities, for
869 example any one of the versions of Universally Unique Identifiers (UUIDs). Context independent
870 identifier may also be generated using hierarchy of domains where the root of the hierarchy is
871 identified with a UUID and sub-domains may generate context independent identifier by
872 concatenating context-specific identifiers for that domain to the context-independent identifier of
873 their parent.

874 6.2.1 Resource identification and addressing

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

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

880 **<scheme>://<Authority>/<Path>?<Query>**

881 Specifically the OIC URI is specified in the following form:

882 **oic://<Authority>/<Path>?<Query>**

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

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

887 The *scheme* above is modified by replacing 'oic' with the indicator for the underlying transport
888 protocol (e.g., 'coap', 'http') before sending over the network by the requestor. Similarly on the
889 receiver side, the indicator for the underlying transport protocol (e.g., 'coap' or 'http') is replaced
890 with 'oic' before handing over to resource model layer on receiver.

891 If the authority is the local OIC Device, then 'oic' shall be used as the authority.

892 The usual form of the authority is

893 **<host>:<port>**, where <host> is the name or endpoint network address and <port> is the network
894 port number. The <host> may be provided as follows:

- 895 • For IP networks, the hostname or IP address of <authority>
- 896 • For non-IP networks, the name or appropriate identifier.
- 897 • If the <authority> is the OIC Device that hosts the resource then the keyword 'oic' may be
898 used for the <host>.

899 The *path* shall be unique string that unambiguously identifies or references a resource within the
900 context of the OIC Server. A *path* shall be preceded by a '/' (slash). The *path* may have '/' (slash)

901 separated segments for human readability reasons. In the OIC context, the '/' (slash) separated
902 segments are treated as a single string that directly references the resources (i.e. a flat structure)
903 and not parsed as a hierarchy. On the OIC Server, the path or some substring in the path may be
904 shortened by using hashing or some other scheme provided the resulting reference is unique within
905 the context of the host.

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

908 A query string shall contain a list of <name>=<value> segments (aka "name-value pair") each
909 separated by a ';' (semicolon). The query string will be mapped to the appropriate syntax of the
910 protocol used for messaging. (e.g., CoAP).

911 A URI may be either

- 912 • Fully qualified or
- 913 • Relative

914 *Generation of URI:*

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

920 *Use of URI:*

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

926 **6.3 Namespace:**

927 The path prefix /oic for relative URIs is reserved as a namespace for OIC defined resources
928 (including discoverable resources). Resources not defined by OIC specifications shall not use
929 prefix /oic for relative URIs.

930 **6.4 Network addressing**

931 The following are the addresses used in this specification:

- 932 • **IP address**

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

934 When an OIC Device only has the identity information of its peer, a resolution mechanism is needed
935 to map the identifier to the corresponding address.

936 7 Resource model

937 7.1 OIC resource model

938 7.1.1 Introduction

939 The OIC resource model provides core interoperability among OIC Devices. This section describes
940 the concepts and mechanisms used to represent the primary details of a resource using the OIC
941 Framework. These include OIC Resource, OIC Interface, OIC Resource Property, OIC Resource
942 Type, resource representation, and resource structures including collections.

943 7.1.2 OIC Resource

944 In the OIC Framework, a physical or software artefact or concept that needs to be made visible
945 and manipulated is represented as an OIC Resource. The OIC Resource encapsulates and
946 represents the salient aspects of an Entity. An OIC Resource has an address as defined in section
947 6, and properties defined in section 7.1.4. In general, an OIC Resource can be assigned any URI
948 and the URI doesn't encode or provide any information of the resource characteristics. However
949 some core OIC Resources have fixed URI (e.g., /oic/res defined in section 11.3.4) and the fixed
950 URI determines its characteristics.

951 An OIC Resource requires that the following properties are specified:

- 952 • **Fixed URI** (optional) – a fixed URI assigned to an OIC Resource for a Resource Type ID (e.g.,
953 /oic/res, /oic/d).
- 954 • **Resource Type Title (optional)** – a human friendly name to designate the resource type.
- 955 • **Resource Type ID** – the value of "rt" property which identifies the Resource Type, (e.g.,
956 oic.r.humidity). A string that has segments separated by a '.' (dot); each segment may represent
957 a name space and in that case later segments (L -> R) would represent sub-name spaces;
958 Implementations shall use these opaquely and use exact string matches – this allows other
959 forms of naming like URNs where required for ecosystem interoperability.
- 960 • **Resource Interfaces** – list of the interfaces that may be supported by the resource type.
- 961 • **Resource Properties** – definition of all the properties that apply to the resource type. The
962 resource type definition shall define whether a property is mandatory, conditional mandatory,
963 or optional.
- 964 • **Related Resource Types** (optional) – the specification of other resource types that may be
965 referenced as part of the resource type, applicable to collections.
- 966 • **Mime Types** (optional) – mime types supported by the resource including serializations (e.g.,
967 application/cbor, application/json, application/xml).

968 Table 2 and Table 3 provide an example description of foobar resource type and properties.

969

Table 2. Example foobar Resource Type

fixed 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

970

Table 3. Example foobar properties

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

971

972 An instance of the foobar resource type is as shown below

973

974

975

976

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

977 An example schema for the foobar resource type is shown below

978

979

980

981

982

983

984

985

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

986 7.1.3 OIC Interface

987 An interface (term derived from IETF RFC 6690) is used to qualify and select the type of permitted
988 requests on an OIC Resource. A resource interface identifies the specific context in which a
989 request is to be evaluated; the same request when evaluated against different resource interfaces
990 would elicit different responses.

991 7.1.4 OIC Resource Properties

992 7.1.4.1 Introduction

993 An OIC Resource Property, or Property in short, describes an aspect or notion that is exposed
994 through the OIC Resource including meta-information related to that resource.

995 Properties can be consumed within the OIC Framework, e.g., a query may use specific Properties
996 with desired values to select OIC Resources that meet those criteria. Properties may also be
997 exposed to the applications.

998 The following items may be included when defining an OIC Property:

- 999 • **Property** – a “key=value” pair
- 1000 • **Property Title** - a human friendly name to designate the property, usually not sent over the
1001 wire.
- 1002 • **Property Name**– the key in "key=value" pair. The data type of the property name is “string”.
- 1003 • **Property Value** – the value in "key=value" pair.
- 1004 • **Value Type** –the data type that the value of the property may take as defined in section 3.4
1005 (e.g., string or boolean).
- 1006 • **Value Rules** –the rules on the data value that the property may take. The rules could for
1007 example define a fixed range, a set of enumerated values, a pattern, conditional values, or
1008 dependencies on other values. The specification of rules could be used by a validator that
1009 validates these values in an implementation (e.g., "enum" : ["oic.if.baseline", "oic.if.ll", "oic.if.b",
1010 "oic.if.rp", "oic.if.p"]).
- 1011 • **Units** – specifies the units that the values for this property are referenced against.
- 1012 • **Access Modes** – specifies whether the value may be read or written to. Updates are equivalent
1013 to a write. R is used for read and W for write – both can be specified (note: write does not
1014 automatically imply read).
- 1015 • **Description** – is descriptive text on the purpose and expected use of this property.
- 1016 • **Mandatory** – specifies if the OIC Property is mandatory or not for a given resource

1017 In general, a property is meaningful only within the resource to which it is associated. However a
1018 base set of properties that may be supported by all OIC Resources, known as common properties,
1019 keep their semantics intact across resources i.e. their “key=value” pair means the same in any
1020 resource. Detailed tables with the above fields for all common properties are defined in section
1021 7.1.4.2.

1022

1023 7.1.4.2 Common properties

1024 7.1.4.2.1 Introduction

1025 The common properties defined in this section may be specified for all OIC Resources. The
1026 following properties are defined as common properties “Resource Type”, “Resource Interface”,
1027 “Policy” and “Name”.

1028 The name of a common property shall be unique and shall not be used by other properties. When
1029 defining a new Resource Type, its non-common properties shall not use the name of existing
1030 common properties (e.g., "rt", "if", "p"). When defining a new "common property", it should be
1031 ensured that its name has not been used by any other properties. The uniqueness of a new
1032 common property name can be verified by checking all the properties of all the existing OIC defined
1033 Resource Types. However, this may become cumbersome as the number of OIC Resource Types
1034 grow. To prevent such name conflicts in the future, OIC may reserve a certain name space for
1035 common property. Potential approaches are (1) a specific prefix (e.g. "oic") may be designated
1036 and the name preceded by the prefix (e.g. "oic.psize") is only for common property; (2) the names
1037 consisting of one or two letters are reserved for common property and all other properties shall
1038 have the name with the length larger than the 2 letters; (3) common properties may be nested
1039 under specific object to distinguish themselves.

1040 **7.1.4.2.2 Resource type**

1041 A resource type indicates a class or category of resources. A resource type may explicitly mandate
1042 the existence of an optional common Property.

1043 The resource type may be pre-defined. A resource type may be defined either by this specification
1044 (called a core resource type), OIC vertical specifications (called a vertical resource type), by a
1045 manufacturer, end user, or developer of OIC Devices (called a vendor defined resource type).

1046 Note: Pre-defined types and details may be communicated out of band (like in documentation) while explicitly defined
1047 resource types may be downloaded and used by, for example, an API or application.

1048 Every OIC Resource shall have a resource type at the time of its creation. Resource types may be
1049 explicitly discovered or knowledge of type may be implicitly shared between the user (e.g. OIC
1050 Client) and the host of the OIC Resource (e.g., OIC Server).

1051 **7.1.4.2.3 Interface**

1052 The Interface property lists the interfaces the resource supports. A resource interface may be
1053 defined either by OIC specifications (a “core resource interface”), OIC vertical specifications (a
1054 “vertical resource interface”), by a manufacturer, end user, or developer of OIC Devices (a “vendor
1055 defined resource interface”).

1056 All resources shall have at least one Interface. The Default Interface of a resource is defined in
1057 section 3.1.31 and if discoverable shall be communicated to the client during resource discovery
1058 in the response to a RETRIEVE operation on '/oic/res'. The Default Interface shall be defined by
1059 an OIC specification.

1060 In addition to any OIC specification defined interface, all OIC resources shall support the Baseline
1061 interface (oic.if.baseline) as defined in Table 8.

1062 Selection of an interface may appear in the query parameter. If no query parameter is specified,
1063 then the System Default Interface is selected.

1064 **7.1.4.2.4 Policy**

1065 Policy defines how the resource is to be accessed or treated. Policy may be tied with security
1066 where required. Two policies are defined:

- 1067 • Discoverable: A resource can be explicitly hidden or made discoverable using this policy. Only
1068 resources marked as discoverable shall be returned as part of a discovery through /oic/res.
- 1069 • Observable: A resource not marked as observable does not allow OBSERVE operation. An
1070 error shall be returned on such requests

1071 Discoverable and Observable are independent of each other and both can be present in the same
1072 OIC Resource.

1073 **7.1.4.2.5 Name**

1074 A human friendly name for the resource, i.e. a specific resource instance name (e.g.,
1075 MyLivingRoomLight),

1076 **7.1.5 Resource representation**

1077 The externally visible and operable snapshot of OIC Resource Properties and their respective
1078 values taken at a point in time is known as the resource representation. Resource representation
1079 captures the state of an OIC Resource at a particular time. The resource representation is
1080 exchanged in the request and response interactions with an OIC Resource. A resource
1081 representation may be used to retrieve or update the state of a resource.

1082 The resource representation shall not be manipulated by the data connectivity protocols and
1083 technologies (e.g., CoAP, UDP/IP or BLE).

1084 7.1.6 Structure

1085 7.1.6.1 Introduction

1086 In many scenarios and contexts, the OIC Resources may have either an implicit or explicit structure
1087 between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The OIC
1088 Framework provides the means to model and map these structures and the relationships among
1089 OIC Resources. The primary building block for resource structures in OIC Framework is the
1090 collection. A collection represents a container, which is extensible to model complex structures.

1091 7.1.6.2 OIC Link

1092 An OIC Link embraces and extends typed web links as specified in IETF RFC 5988 as a means of
1093 expressing relationships between resources. An OIC Link consists of:

- 1094 • a context IRI,
- 1095 • a target IRI,
- 1096 • a relation from the context IRI to the target IRI
- 1097 • a set of parameters that provide metadata about the target IRI.

1098 The target IRI is mandatory and the other items in an OIC Link are optional. Additional items in the
1099 OIC Link may be made mandatory based on the use of the links in different contexts (e.g. in
1100 collections, in discovery, in bridging etc.).

1101 IETF RFC 5988 establishes a registry for link relations (section 4.1 of IETF RFC 5988) and also
1102 defines a base set of attributes (section 5.4 of IETF RFC 5988). Furthermore, IETF RFC 6690
1103 utilizes typed web links to enable discovery of hosted resources within Constrained RESTful
1104 Environments (CoRE). It also adds a number of attributes (Section 3 of IETF RFC 6690) to allow
1105 typed web links to contain metadata that is useful for resource discovery.

```
1106 {  
1107   {  
1108     "$schema": "http://json-schema.org/draft-04/schema#",  
1109     "id": "http://openinterconnect.org/rm/oic.oic-link.json",  
1110     "type": "object",  
1111     "properties": {  
1112       "href": {  
1113         "type": "string",  
1114         "description": "RFC5988 style web-links described using JSON. This is the target URI",  
1115         "format": "uri"  
1116       },  
1117       "rel": {  
1118         "type": "string",  
1119         "default": "null",  
1120         "description": "The relation of the target URI referenced by the link to the context URI; the reserved  
1121 value of 'null' is for no relationship"  
1122       },  
1123       "rt": {  
1124         "type": "string",  
1125         "description": "Resource Type - A standard OIC specified or vendor defined resource type of the resource  
1126 referenced by the target URI"  
1127       },  
1128       "if": {  
1129         "type": "string",  
1130         "description": "Interface - The interfaces supported by the resource referenced by the target URI"  
1131       },  
1132       "obs": {  
1133         "type": "boolean",  
1134         "description": "Specifies if the resource referenced by the target URI is observable or not",  
1135         "default": false  
1136       },  
1137       "title": {  
1138         "type": "string",  
1139         "description": "A title for the link relation. Can be used by the UI to provide a context"  
1140       }  
1141     }  
1142   }  
1143 }
```

```

1140     },
1141     "anchor": {
1142       "type": "string",
1143       "description": "This is used to override the context URI e.g. override the URI of the containing collection",
1144       "format": "uri"
1145     },
1146     "ins": {
1147       "oneOf": [
1148         {
1149           "type": "integer",
1150           "description": "An ordinal number that is not repeated - must be unique in the collection context"
1151         },
1152         {
1153           "type": "string",
1154           "description": "Any unique string including a URI"
1155         },
1156         {
1157           "type": "string",
1158           "format": "uuid",
1159           "description": "Use UUID for universal uniqueness - used in /oic/res to identify the device"
1160         }
1161       ],
1162       "description": "The instance identifier for this web link in an array of web links - used in collections"
1163     },
1164     "mt": {
1165       "type": "array",
1166       "description": "A hint of the media type of the representation of the resource referenced by the target
1167 URI",
1168       "items": {
1169         "type": "string"
1170       },
1171       "minItems": 1,
1172       "default": "[ application/cbor ]"
1173     },
1174   },
1175   "required": [
1176     "href",
1177     "rt",
1178     "if"
1179   ]
1180 }

```

1181 **Figure 7: JSON Schema for OIC Links**

1182 As shown in Figure 7 the relation between the context IRI and target IRI is specified using the “rel”
1183 JSON element and the value of this element specifies this relation. If a web link does not explicitly
1184 include the “rel” element, a value of “rel”=“hosts” shall be assumed for all web links returned in
1185 response to the “/oic/res” resource. The relation value of “hosts” is defined by IETF RFC 6690 and
1186 registered in the IANA Registry for Link Relations at [<http://www.iana.org/assignments/link-relations/link-relations.xhtml>]

1187
1188 The anchor parameter is used to change the context URI of an OIC Link – the relationship with the
1189 target URI is based off the anchor URI when the anchor is specified. An example of using anchors
1190 in the context of OIC Collections – a floor has rooms and rooms have lights – the lights can be
1191 defined in floor as OIC Links but the OIC Links will have the anchor set to the URI of the rooms
1192 that contain the lights (the relation is contains). This allows all lights in a floor to be turned on or
1193 off together while still having the lights defined with respect to the rooms that contain them (lights
1194 can also be turned on by using the room URI too).

```

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

```

```

/a/room1 {
  "links": [
    {
      "href": "/x/light1",
      "rel": "contains"
    }
  ]
}

```

1195 **Figure 8: Example of use of anchor in OIC Link**

1196

1197 The OIC architecture utilizes typed web links as a mechanism for bootstrapping resource discovery
1198 through the known OIC Core Resource "/oic/res". Additionally for this specification the IRIs are
1199 limited to URIs. A RETRIEVE operation on "/oic/res" returns (among other things) a serialized
1200 representation of typed web links to resources that are discoverable from that OIC Device. The
1201 serialization format should be negotiated using the underlying transport protocol i.e. using Accept
1202 and Content-Type headers in case of HTTP and CoAP. By default OIC uses CBOR as the payload.
1203 The payload (content) in CBOR for web links is described with the JSON Schema in Figure 7.
1204 Other serializations (e.g., XML/EXI) may be defined in future versions of this specification.

1205

1206 Figure 9 is a JSON Schema that specifies the representation of the response to "/oic/res"

```

1207 {
1208   "$schema": "http://json-schema.org/draft-v4/schema#",
1209   "id": "http://openinterconnect.org/schemas/oic.res.json/",
1210   "definitions": {
1211     "oic.res-links.json": {
1212       "type": "object",
1213       "properties": {
1214         "di": {
1215           "description": "The device identifier as indicated by the /oic/d resource of the device",
1216           "type": "string",
1217           "format": "UUID"
1218         },
1219         "links": {
1220           "type": "array",
1221           "items": {
1222             "$ref": "oic.oic-link.json#"
1223           }
1224         }
1225       }
1226     },
1227   },
1228   "description": "The list of resources expressed as OIC Links",
1229   "type": "array",
1230   "items": {
1231     "$ref": "#/definitions/oic.res-links.json"
1232   }
1233 }

```

1234

1235 **Figure 9: JSON Schema for /oic/res**

1236

1237 Figure 10 is an example of a RETRIEVE from "/oic/res"

```

1238 [
1239   {
1240     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1241     ... "links": [
1242       {
1243         "href": "/door",
1244         "rt": "oic.r.door",
1245         "if": "oic.if.b oic.ll"
1246       },
1247       {
1248         "href": "/door/lock",
1249         "rt": "oic.r.lock",
1250

```

```

1251     "if": "oic.if.b",
1252     "type": "application/cbor application/exi+xml"
1253   }
1254 ]
1255 },
1256 {
1257   "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
1258   "links": [
1259     {
1260       "href": "/light",
1261       "rt": "oic.r.light",
1262       "if": "oic.if.s"
1263     },
1264     {
1265       "href": "/binarySwitch",
1266       "rt": "oic.r.switch.binary",
1267       "if": "oic.if.a",
1268       "type": "application/cbor"
1269     }
1270   ]
1271 }
1272 ]

```

1273 **Figure 10: Example /oic/res representation**

1274

1275 The “type” element may be used to specify the various representations that are supported by a
1276 specific resource. The default type of application/cbor shall be used when “type” element is omitted.
1277 The example in Figure 10 shows that the “/door/lock” resource supports two representations (cbor
1278 and exi/xml); but is described in json for readability. Additional representations may be included
1279 as necessary. Once a client discovers this information for each resource, it may use one of the
1280 available representations in the Accept header of the request.

1281 **7.1.6.3 Collections**

1282 **7.1.6.3.1 Overview**

1283 An OIC Resource that contains one or more references (specified as OIC Links) to other resources
1284 is an OIC Collection. These reference may be related to each other or just be a list; the OIC
1285 Collection provides a means to refer to this set of references with a single handle (i.e. the IRI). A
1286 simple resource is kept distinct from a collection. Any OIC Resource may be turned into an OIC
1287 Collection by binding resource references as OIC Links. OIC Collections may be used for creating,
1288 defining or specifying hierarchies, indexes, groups, and so on.

1289 An OIC Collection shall have at least one resource type and at least one interface bound at all
1290 times during its lifetime. A resource type and interfaces shall be bound to a collection at the
1291 creation of the collection. These initial values may be overridden using mechanism used for
1292 overriding in the case of an OIC Resource. Additional resource types and interfaces may be bound
1293 to the collection at creation or later during the lifecycle of the OIC Collection.

1294 An OIC Collection shall define the “links” common property. The value of the “links” property is an
1295 array with zero or more OIC Links. The target URIs in the OIC Links may reference another OIC
1296 Collection or another OIC Resource. The referenced OIC Collection or OIC Resource may reside
1297 on the same OIC Device as the OIC Collection that includes that OIC Link (called a local reference)
1298 or may reside on another OIC Device (called a remote reference). The context URI of the OIC
1299 Links in the “links” array shall (implicitly) be the OIC Collection that contains that “links” property.
1300 The (implicit) context URI may be overridden with explicit specification of the “anchor” parameter
1301 in the OIC Link where the value of “anchor” is the new base of the OIC Link.

1302 An OIC Resource may be referenced in more than one OIC Collection, therefore, a unique parent-
1303 child relationship is not guaranteed. There is no pre-defined relationship between a collection and
1304 the resource referenced in the collection, i.e., the application may use collections to represent a
1305 relationship but none is automatically implied or defined. The lifecycles of the collection and the
1306 referenced resource are also independent of one another.

1307 If the “drel” property is defined for the collection then all OIC Links that don’t explicitly specify a
1308 relationship shall inherit this default relationship in the context of that collection. The default
1309 relationship defines the implicit relationship between the collection and the target URI in the OIC
1310 Link.

1311 The list of OIC Links defined in a collection may be either a simple list of OIC Links (i.e. the value
1312 of the “links” property is a simple array of OIC Links) as illustrated in Figure 12 or may be a list of
1313 tagged sets of OIC Links where each tagged set shall have at least one tag (i.e. the value of the
1314 “links” property is an array where each element of the array is an object with an array of OIC Links
1315 and a set of one or more key-value pairs; the key-value pairs in that object are the tags for the
1316 array of OIC Links in that object; the key is the tag name and the value is the tag value) as
1317 illustrated in Figure 13.



1318
1319 Figure 11: Example showing parts of OIC Collection and OIC Links

```
1320 {
1321   "links": [
1322     {
1323       "href": "/door",
1324       "rt": "oic.r.door",
1325       "if": "oic.if.b oic.ll"
1326     },
1327     {
1328       "href": "/door/lock",
1329       "rt": "oic.r.lock",
1330       "if": "oic.if.b",
1331       "type": "application/cbor application/exi+xml"
1332     }
1333   ]
1334 }
```

```
}
]
}
```

1321

Figure 12: Example OIC Collection with simple links (JSON)

1322

```
{
  "links": [
    [
      {
        "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
      },
      [
        {
          "href": "/door",
          "rt": "oic.r.door",
          "if": "oic.if.b oic.ll"
        },
        {
          "href": "/door/lock",
          "rt": "oic.r.lock",
          "if": "oic.if.b",
          "type": "application/cbor application/exi+xml"
        }
      ]
    ],
    [
      {
        "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
      },
      [
        {
          "href": "/light",
          "rt": "oic.r.light",
          "if": "oic.if.s"
        },
        {
          "href": "/binarySwitch",
          "rt": "oic.r.switch.binary",
          "if": "oic.if.a",
          "type": "application/cbor"
        }
      ]
    ]
  ]
}
```

1323

Figure 13: Example OIC Collection with tagged OIC Links (JSON)

1324 Note: Example shows only one tag; each tag has the same tag name, i.e., "di", but have different tag values.

1325

1326 A collection may be:

- 1327 • A pre-defined collection where the collection has been defined a priori and the collection is
1328 static over its lifetime. Such collections may be used to model, for example, an appliance that
1329 is composed of other devices or fixed set of resource representing fixed functions.
- 1330 • A device local collection where the collection is used only on the OIC Device that hosts the
1331 collection. Such collections may be used as a short-hand on a client for referring to many
1332 servers as one.
- 1333 • A centralized collection where the collection is hosted on an OIC Device but other OIC Devices
1334 may access or update the collection
- 1335 • A hosted collection where the collection is centralized but is managed by an authorized agent
1336 or party.

1337 7.1.6.3.2 Collection properties

1338 An OIC Collection shall define the "links" property. In addition, other properties may be defined for
1339 the collection by the resource type. The mandatory and recommended common properties for

1340 collection are shown in Table 4. This list of common properties are in addition to those defined for
 1341 OIC Resources in section 7.2.1. When a property is repeated in Table 4 for OIC Collections, the
 1342 conditions in this definition shall override those in the general list for OIC Resources.

1343 **Table 4: Common properties for OIC Collections (in addition to Common Properties**
 1344 **defined in section 7.2.1)**

Property	Description	Property name	Value Type	Mandatory
Links	The set of links in the collection	"links"	json Array of OIC 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"	CSV Format: Comma separated list of resource type names	No
Default relationship	Specifies the default relationship to use for OIC 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 OIC 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	"rel"	string	No

1345
 1346 The properties of a collection may be modified by using the UPDATE (POST) operation to the
 1347 oic.wk.baseline interface with a jSON object having the list of properties and their updated values.
 1348 In the case of a single property being updated, the query with property name could be used to
 1349 select the property and the request payload has the value of that property (e.g. GET /<col
 1350 URI>?property).

1351 **7.1.6.3.3 Interfaces for collections**

1352 The interfaces allows for modulation of the requests against collection for a given resource type.
 1353 The primary interfaces defined by OIC are shown in Table 5.

1354 The resource type determines which of these interfaces are valid for that collection type. For
 1355 example: a factory collection (defined by a resource type that includes factory semantics) may
 1356 support the "link batch" interface but a resource type for a simple collection may only define a
 1357 "links list" interface.

Table 5: OIC-defined Interfaces for OIC Collection

Interface	Name	Description
Baseline	oic.if.baseline	Interface provides access to the entire representation of the collection. Can be used to modify properties and links in the collection.
Links List	oic.if.ll	Interface provides access to only the links in the collection i.e. the value of the “links” property. In the case of tagged links, the tag of the link shall be provided in the query. A RETRIEVE using this interface retrieves the list of links A UPDATE on this interface can be used to add, modify or update the links in the collection. . When UPDATE is done using this interface, no checks are done if the links reference valid and available resources. This interface is used to simply “update” the links.
Batch	oic.if.b	Interface provides access to the resources/collections that are referenced by the links in the collection. A RETRIEVE using this interface shall return the representation of all the resources referenced by links in the collection. The response that have all the representation aggregated. A UPDATE on this interface shall lead to a UPDATE request to each of the referenced resources in the collection’s links; the representation in the payload for each of these implicit requests is the same as the request to the collection. Properties in the payload that are not understood by the target resources shall be ignored. [NOTE: Security implications of using this interface will be specified in the future when security interactions are defined beyond link layer security].
Link Batch	oic.if.lb	Interface provides means to create the resource that is reference in a link or set of links and add those links to the collection in one atomic operation. (In this instance, atomic implies that no new requests are processed against this collection until this request and response completes fully or fails). An UPDATE request on this interface will carry the link(s) that need to have resource created and then added to the collection. If the resource reference by the link is already present then the link is tested to be valid and available. [NOTE: This is a harder implication on the updates on links and should be used carefully and judiciously only when this behaviour is necessary. Simple operations on links can be done using update with the “links list” interface] A RETRIEVE request on this interface is not supported. [NOTE: Security implications of using this interface will be specified in the future when security interactions are defined beyond link layer security].

1359 **7.1.6.3.4 Default resource type**

1360 A default resource type, oic.wk.col, shall be available for OIC Collections. This resource type shall
1361 be used only when another type has not been defined on the collection or when no resource type
1362 has been specified at the creation of the collection.

1363 The default resource type provides support for the common properties in Table 5 including the
1364 “links” property. For the default resource type, the value of “links” shall be a simple array of OIC
1365 Links and tagging of links shall not be supported.

1366 The default resource type shall support the ‘baseline’ and ‘links list’ interfaces.

1367 **7.1.6.3.5 Bootstrapping collections**

1368 OIC Devices that support creation of collections shall expose the standard bootstrap resource
1369 /oic/col with resource type oic.wk.col and supporting the link batch interface oic.if.lb. The bootstrap
1370 resource shall be discoverable and observable.

1371 The resource /oic/col shall be discovered through /oic/res to indicate to other OIC Device that
1372 collection creation is supported.

1373 Once the bootstrap resource has been discovered, an OIC Client may request creation of
1374 collections on that bootstrap resource. This is done by sending a CREATE (POST) operation to
1375 the oic.if.lb interface of /oic/col with a body having a representation defined by the default resource
1376 type oic.wk.col. The links in the body shall have the resource type of the collection or resources to
1377 be created. More than one collection may be requested and created in a single request. The policy
1378 for the created collection may be defined in the OIC Link for that collection (by default the policy
1379 for collections created using /oic/col is discoverable and observable). The target URIs in the links
1380 in the request shall have either a URI as a hint to the OIC Server or have the nul value to indicate
1381 that the server is to generate. The OIC Server may ignore the URI hint.

1382 The result of the request is that the collections in the request are created and, if the policy
1383 “discoverable” does not disallow, will be published in /oic/res in one atomic operation. The links
1384 with the URIs of the created collection shall be returned in the response. The OIC Server shall
1385 also set the value of the “ins” parameter of the links returned in the response.

1386 **7.1.6.3.6 Creation of collection and resource using collections**

1387 An OIC Collection that exposes the link batch interface can be used as a factory for resources of
1388 the resource types that the collection permits (see “rts” property). The representation in the body
1389 of the request shall be defined by the resource type of the collection to which the request is sent.

1390 The creation of the resource or collection and the information returned in the response is similar
1391 to the bootstrap. The OIC Links for the created collection or resource may be added to the OIC
1392 Collection that process the request.

1393 The default policy for resource created through general collections (unlike bootstrap) is non-
1394 discoverable and non-observable.

1395 **7.1.6.3.7 Deletion of collections**

1396 To delete a collection a DELETE operation is sent to the URI of the collection to be deleted.
1397 Optionally, a policy shall be defined on the collection to disallow deletion if the collection has links.
1398 If the policy disallows deletion of a collection with links then an error shall be returned.

1399 To delete a collection and all the collections and resource that are referenced from that collection
1400 then the DELETE operation shall be sent to the link batch interface of the collection (e.g., DELETE
1401 /<coll URI>?if=oic.if.lb). This is a dangerous operation and should be used with caution.

1402 **7.1.6.3.8 Managing collections**

- 1403 • Add links:

1404 To add links into the collection, a UPDATE (POST) operation to the collection URI shall use
1405 the oic.if.ll interface in the query. The body of the request shall have the “links” property and
1406 the value shall be an array of one or more OIC Links to add to the collection. (No checking is
1407 done on the OIC Server if the URIs in the OIC Link point to valid and active resources). For a
1408 add request the “ins” parameter in the links in the request payload shall be empty/null or the
1409 “ins” parameter shall be omitted.

1410 The OIC Server shall respond to the request by adding the links to the collection and returning
1411 in the body of the response the OIC Links with the “ins” parameter assigned to the appropriate

1412 value. This value of the “ins” parameter shall uniquely identify this link within the collection
1413 over the entire lifetime of the collection.

1414 The payload of the response shall also include the Time to Live for the links – a single time to
1415 live is support for all links in the request payload.

1416 • **Modify links**

1417 To modify the links in a collection, a UPDATE (POST) operation to the collection URI shall use
1418 the oic.if.ll interface in the query. The body of the request shall have the “links” property and
1419 the value shall be an array of one or more OIC Links to add to the collection. For a modify
1420 request, the “ins” parameter shall have the value that identifies the link to be modified. (No
1421 checking is done on the OIC Server if the URIs in the OIC Link point to valid and active
1422 resources).

1423 The OIC Server shall replace the parameters of link in the collection that has the matching “ins”
1424 value with corresponding parameters in the link in the request payload. To replace the entire
1425 link all link parameters (except “ins”) in the collection have to have updated values. This method
1426 may be used to update only the Time to Live for the links.

1427 • **Delete links**

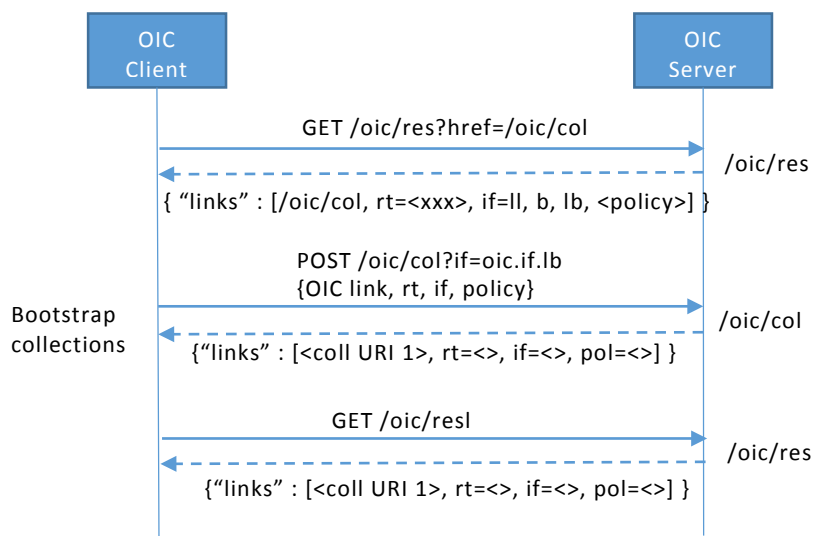
1428 To delete links in a collection, a DELETE (DELETE) operation is sent to collection URI with the
1429 instance to delete identified in the query of the request (e.g. DELETE /<coll URI>?
1430 if=oic.wk.ll;ins=”yyz”).

1431 For multiple instance to be deleted, the “ins” parameter may be repeated for each of the link
1432 (interface oic.wk.ll is specified only once).

1433 To delete all the links in a collection, the DELETE operation shall be sent to the link list interface
1434 of the collection (e.g. DELETE /<coll URI>?if=oic.wk.ll)

1435 **7.1.6.3.9 Example flows**

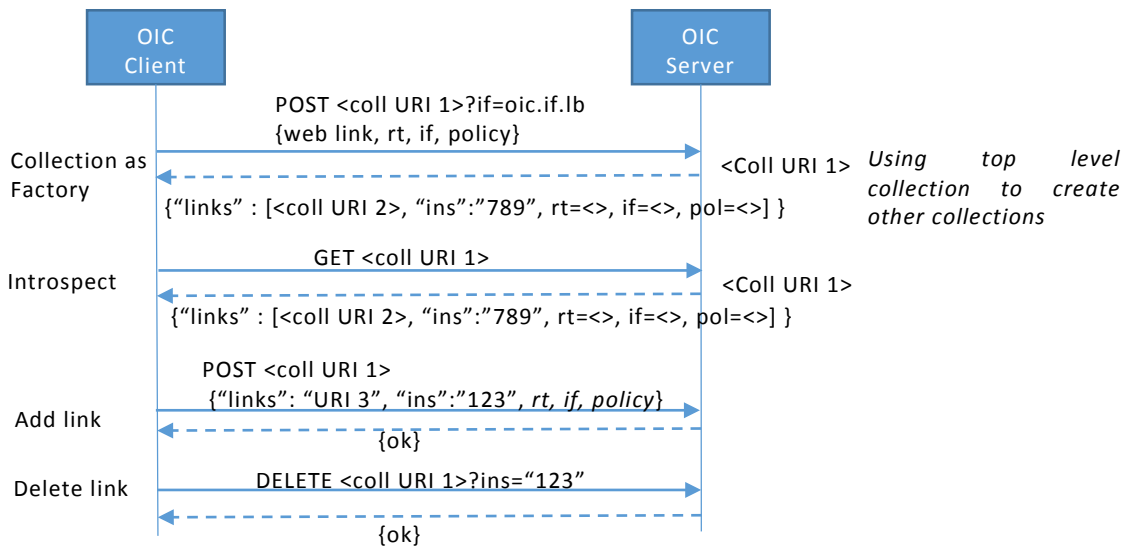
1436 Figure 14 and Figure 15 provides examples of message flows for managing collections



1437

1438

Figure 14. Bootstrap collections



1439

1440

Figure 15. Collection as a factory

1441 **7.1.6.3.10 Selectors and filters (in query strings)**

1442 In many use cases, there is a need to return a handle (URI) to a collection based on the matching
 1443 of the properties and parameters of the links in collection. For example: trying to get all rooms in
 1444 a floor that have LED bulbs In these cases, a simple query is insufficient because the properties
 1445 of a collection can also appear in the links in the collection – there is no clean way to disambiguate
 1446 using the simple query mechanism. OIC introduces the filter keyword in the query to separate the
 1447 part of the query – one on the properties of the collection itself and one on the properties in the
 1448 links in the collection. The selector selects based on the properties themselves whereas the filter
 1449 is based key-value pairs in the values of properties in the collection.

1450 So GET /oic/res?rt=oic.wk.foo will match all collections in /oic/res that are of resource type
 1451 oic.wk.foo. A GET /oic/res?filter="rt=oic.wk.foo" will match collections in /oic/res that *contain*
 1452 collections that are of resource type oic.wk.foo. (Using an interface to separate selectors and filter
 1453 is not possible since query strings don't have an ordering in the key value pairs).

1454 A query in a request shall have two parts – a selector which is the primary query and a filter. The
 1455 filter shall be designated with the keyword "filter". The value of the filter is also a query with the
 1456 same key-value pairs in query without the filter keyword. A filter shall be applied to match the query
 1457 in the filter against the links in the collection – if a match is found then the collection containing
 1458 the match shall be used as the resulting URI of the filtered query. A query string not enclosed in a
 1459 "filter" is a selector and shall be applied to match the properties-values of the collection itself.

1460

1461 **7.2 Usage of OIC resource model**

1462 **7.2.1 Values for common properties**

1463 All OIC Resources shall have the following common properties as defined in section 7.2.1.1
 1464 through section 7.2.1.4.

1465 **7.2.1.1 Resource type property definition**

1466 **Table 6. Resource type property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Resource type	rt	string	The resource type ID		R	yes	The property name rt is as described in IETF RFC 6690

1467 Resource type property name (Table 6) 'rt' shall be designated as 'rt=<resource type value>'.
 1468 The value of a resource type property is a '.' (dot) separated string where each sub-string
 1469 separated by a "." represents a namespace and the last sub-string represents the name of the
 1470 resource type in that namespace hierarchy. The namespace 'oic' when used in the first sub-
 1471 string, is reserved for OIC specifications and is used to namespace OIC Resource types. The
 1472 resource type value may also be a reference to an authority similar to IANA that may be used to
 1473 find the definition of a resource type.

1474 Every OIC Resource shall have at least one resource type. A single resource may be assigned
 1475 multiple resource types only when the assigned resource types don't conflict. The resource type is
 1476 bound to a resource at the time of its creation.

1477 Some examples follow:

- Light is resource type (one type to manage both toggle and dimmer); client may only read properties
/light/a, rt=Light, if= oic.if.r
 - *ToggleLight* is a resource type that defines the "powerstate" property which may have value "on"/"off"; may set and read on this resource
/light/b, rt=ToggleLight, if= oic.if.rw
 - Using this URI client may set and read the "dimmerstate" property (defined in resource type *DimmerLight*)
/light/c, rt=DimmerLight, if= oic.if.rw
 - Since the resource type "Light" supports read and set both powerstate and dimmerstate, the URI '*/light/d*' may be used to turn on/off the light and also to dim it.
/light/d, rt=Light, if= oic.if.rw
- All the URI refer to the same light bulb. The property name 'rt' defines the resource type and 'if' defines the interface*

1478 **7.2.1.2 Resource interface property definition**

1479 **Table 7. Resource interface property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface	if	string	Dot separated string		R	yes	The property name if is as described in IETF RFC 6690

1480 Resource interface property name (Table 7) 'if' shall be designated as 'if=<resource interface
 1481 value>'. The value of a resource interface property is a '.' (dot) separated string where each sub-
 1482 string separated by a "." represents a namespace and the last sub-string represents the name of
 1483 the resource interface in that namespace hierarchy. The namespace 'oic' when used in the first
 1484 sub-string, is reserved and is used to namespace OIC Resource interfaces. The resource interface
 1485 value may also be a reference to an authority similar to IANA that may be used to find the definition
 1486 of a resource interface.

1487

```
if=oic.if.ll – Link list interface may return links (reference to other resources).
GET //oic/mycollection?if=oic.if.ll (returns links which are fully qualified)
```

1488

1489 The OIC resource model defines the standard interfaces listed in Table 8.

1490

Table 8. OIC standard interface

Interface	Name	Applicable Methods	Description
Baseline	oic.if.baseline	RETRIEVE, UPDATE	<p>The Baseline interface shall be available on all OIC resources from the time of their creation.</p> <p>The Baseline interface is used when an OIC Client needs to retrieve all the properties of the resource. The OIC Client includes the <code>?if="oic.if.baseline"</code> in a RETRIEVE request. The OIC Server shall respond with a representation of all the properties it has at the time of the request.</p> <p>If the OIC Server is unable to send back the whole resource representation, it shall reply with an error message. The OIC Server shall not return a partial resource representation.</p> <p>The properties of a resource may be modified by using the UPDATE request to the oic.if.baseline interface with the list of properties and their updated values.</p>
List Links	oic.if.ll	RETRIEVE	<p>The List Links interface is used to list the references (links) to other OIC Resources contained in an OIC Resource. The concept borrows from IETF draft-ietf-core-interfaces-02 section 5.1 with the following semantics</p> <p>The serialization shall be in the format requested in the request.</p> <p>In response to a RETRIEVE request on List Links interface, the URIs of the referenced OIC Resources shall be returned as fully qualified URIs.</p> <p>If there are no links present in an OIC Resource then an empty list shall be returned.</p>
Batch	oic.if.b	RETRIEVE, UPDATE	<p>The concept borrows from IETF draft-ietf-core-interfaces-02 section 5.2 with the following semantics</p> <p>An OIC Resource with a Batch interface may have both local and remote references (fully qualified).</p> <p>For local resources, the semantics in the IETF spec applies with the exception that resources shall not use Batch interface as an extension of the Link List interface.</p> <p>For remote resources:</p> <p>A request to the batch interface of an OIC Resource is processed by the OIC Resources that are referenced within that resource. To select a subset of the referenced resources the other query parameters, if present shall be processed first. Then the original request is modified to create new requests targeting each referenced resource by substituting the Authority and Path part of the URI in the original request with the Authority and Path of the referenced resource. The payload in the original request is replicated as the payload for the new request. The response from all the referenced OIC Resources shall be aggregated as a single response to the original request and returned to the OIC Client. If</p>

			the target OIC Resources cannot process the new request, an empty response shall be returned.
Read-Only	oic.if.r	RETRIEVE	The Read-Only interface limits the applicable methods that can be applied to an OIC Resource to RETRIEVE only. An attempt by an OIC Client to apply a method other than RETRIEVE to an OIC Resource when applying the oic.if.r interface shall be rejected with a response code of 4.05 (if using CoAP) or 405 (if using HTTP).
Read-Write	oic.if.rw	RETRIEVE, UPDATE	The Read-Write interface limits the applicable methods that can be applied to an OIC Resource to RETRIEVE and UPDATE only.
Actuator	oic.if.a	CREATE, RETRIEVE, UPDATE	The Actuator interface adheres to the definition of the Actuator interface defined by the IETF draft-ietf-core-interfaces-02 with the following normative changes: A RETRIEVE action on this interface only provides (subject to any queryParameters that may also exist) the properties identified as part of the resource representation; it does not provide the properties defined as common properties for the resource. An UPDATE action on this interface shall also provide a payload or body that contains the requested values for the target resource. A resource using this interface shall return a media type of CBOR as defined in IETF RFC 7049.
Sensor	oic.if.s	RETRIEVE	The Sensor interface adheres to the definition of the Sensor interface defined by the IETF draft-ietf-core-interfaces-02 with the following normative changes: A RETRIEVE action on this interface only provides (subject to any queryParameters that may also exist) the properties identified as part of the resource representation; it does not provide the properties defined as common properties for the resource. A resource using this interface shall return a media type of CBOR as defined in IETF RFC 7049.

1491

1492 The following is an example of the RETRIEVE operation on a batch interface

<p>1. Retrieve with batch interface</p> <p>REQUEST: GET oic://192.168.1.114:52383/a/room?if=oic.if.b</p> <p>RESPONSE: {"oic":[{"href":"/a/room"}, {"href":"/a/light", "rep":{"state":"0", "color":"0"}}, {"href":"/a/fan", "rep":{"state":"1", "speed":10}}]}</p> <p>2. Update with batch interface</p> <p>REQUEST: POST oic:// 192.168.1.114:52383/a/room?if=oic.if.b ; PAYLOAD: {"state":"1", "speed":"1"}</p> <p>RESPONSE: {"oic":[{"href":"/a/room"}, {"href":"/a/light", "rep":{"state":"1", "color":"0"}}, {"href":"/a/fan", "rep":{"state":"1", "speed":1}}]}</p> <p><i>Note: Since both /a/light and /a/fan have a property "state" both these are set to true because of the batch interface.</i></p>

1493

1494 7.2.1.3 Policy property definition

1495

Table 9. Policy Property Definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Policy	p	string	The policy name		R	no	

1496

1497 Policy property name (Table 9) ‘p’ shall be designated as ‘p=<policy value>’. The value of a policy
1498 property is defined as {“bm”: “<bitmap>”} where <bitmap> is a bitmap of one octet.

1499 The least significant bit (LSB) in the bitmap shall be set to 1 if the resource is discoverable and
1500 shall be set to 0 otherwise. The second LSB shall be set to 1 if the resource is observable and set
1501 to 0 otherwise. The default value of any of these specified bits is 0 (which implies that resources
1502 are non-discoverable unless made discoverable and resources are not observable until explicitly
1503 made observable). Not including the Policy property is equivalent of the value of the bitmap being
1504 set to 0.

1505 The setting in the bitmap on the resource will lead to appropriate elements being defined in the
1506 discovery information of that resource and representation of the resource as required.

1507 The bits in the bitmap that are not defined in this specification are reserved for future use of OIC
1508 specifications.

1509

1510 7.2.1.4 Name property definition

1511

Table 10. Name Property Definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R	no	Human understandable name for the resource; may be set locally or remotely (e.g., by a user)

1512

1513 7.2.2 OIC Core Resources

1514 OIC Core Resources are the resources defined in this specification to enable functional
1515 interactions as defined in section 10, e.g., Discovery, Device Management etc. Among the OIC
1516 Core Resources, ‘/oic/res’ and ‘oic/d’ shall be supported on all OIC Devices. OIC Devices may
1517 support other OIC Core Resources depending on the functional interactions they support.
1518 Additionally, a Resource Type ID for all well-known resources is provided for consistency, but is
1519 not transmitted by an OIC Device.

1520

1521 **8 CRUDN**

1522 **8.1 Overview**

1523 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for
1524 manipulating OIC Resources. These operations are performed by an OIC Client on the resources
1525 contained in an OIC Server.

1526 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in
1527 Table 11. An OIC Device shall use CBOR as the default payload (content) encoding scheme for
1528 resource representations included in CRUDN operations and operation responses; an OIC Device
1529 may negotiate a different payload encoding scheme (e.g, see in section 12.2.3 for CoAP
1530 messaging). The following subsections specify the CRUDN operations and use of the parameters.
1531 The type definitions for these terms will be mapped in the messaging section for each protocol.

1532 **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 OIC 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.

1533 **8.2 CREATE**

1534 The CREATE operation is used to request the creation of new OIC Resources on the OIC Server.
1535 The CREATE operation is initiated by the OIC Client and consists of three steps, as depicted in
1536 Figure 16 and described below.

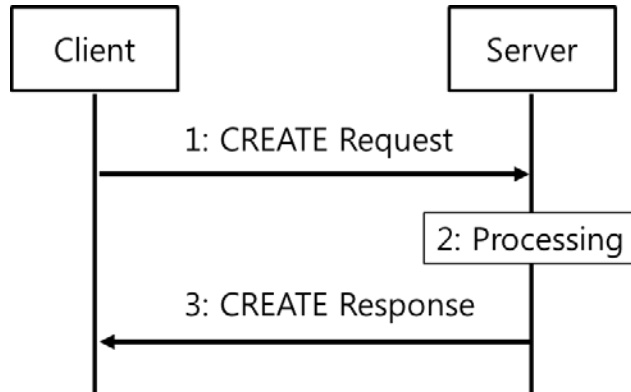


Figure 16. CREATE operation

1537

1538

1539 8.2.1 CREATE request

1540 The CREATE request message is transmitted by the OIC Client to the OIC Server to create a new
 1541 OIC Resource by the OIC Server. The CREATE request message will carry the following
 1542 parameters:

- 1543 • *fr*: Unique identifier of the OIC Client
- 1544 • *to*: URI of the target resource responsible for creation of the new resource.
- 1545 • *ri*: Identifier of the CREATE request
- 1546 • *cn*: Information of the resource to be created by the OIC Server
 - 1547 i) *cn* will include the URI and resource type property of the resource to be created.
 - 1548 ii) *cn* may include additional properties of the resource to be created.
- 1549 • *op*: CREATE

1550 8.2.2 Processing by the OIC Server

1551 Following the receipt of a CREATE request, the OIC Server may validate if the OIC Client has the
 1552 appropriate rights for creating the requested resource. If the validation is successful, the OIC
 1553 Server creates the requested resource. The OIC Server caches the value of *ri* parameter in the
 1554 CREATE request for inclusion in the CREATE response message.

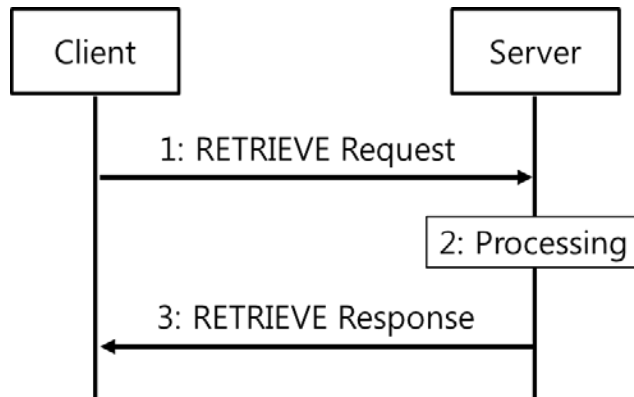
1555 8.2.3 CREATE response

1556 The OIC Server shall transmit a CREATE response message in response to a CREATE request
 1557 message from an OIC Client. The CREATE response message will include the following
 1558 parameters.

- 1559 • *fr*: Unique identifier of the OIC Server
- 1560 • *to*: Unique identifier of the OIC Client
- 1561 • *ri*: Identifier included in the CREATE request
- 1562 • *cn*: Information of the resource as created by the OIC Server.
 - 1563 i) *cn* will include the URI of the created resource.
 - 1564 ii) *cn* will include the resource representation of the created resource.
- 1565 • *rs*: The result of the CREATE operation

1566 **8.3 RETRIEVE**

1567 The RETRIEVE operation is used to request the current state or representation of an OIC Resource.
1568 The RETRIEVE operation is initiated by the OIC Client and consists of three steps, as depicted in
1569 Figure 17 and described below.



1570

1571 **Figure 17. RETRIEVE operation**

1572 **8.3.1 RETRIEVE request**

1573 RETRIEVE request message is transmitted by the OIC Client to the OIC Server to request the
1574 representation of an OIC Resource from an OIC Server. The RETRIEVE request message will
1575 carry the following parameters.

- 1576 • *fr*: Unique identifier of the OIC Client
- 1577 • *to*: URI of the resource the OIC Client is targeting
- 1578 • *ri*: Identifier of the RETRIEVE request
- 1579 • *op*: RETRIEVE

1580 **8.3.2 Processing by the OIC Server**

1581 Following the receipt of a RETRIEVE request, the OIC Server may validate if the OIC Client has
1582 the appropriate rights for retrieving the requested data and the properties are readable. The OIC
1583 Server caches the value of *ri* parameter in the RETRIEVE request for use in the response.

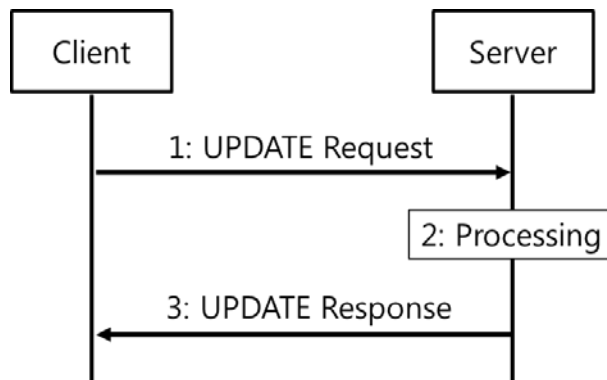
1584 **8.3.3 RETRIEVE response**

1585 The OIC Server shall transmit a RETRIEVE response message in response to a RETRIEVE
1586 request message from an OIC Client. The RETRIEVE response message will include the following
1587 parameters.

- 1588 • *fr*: Unique identifier of the OIC Server
- 1589 • *to*: Unique identifier of the OIC Client
- 1590 • *ri*: Identifier included in the RETRIEVE request
- 1591 • *cn*: Information of the resource as requested by the OIC Client
 - 1592 i) *cn* should include the URI of the resource targeted in the RETRIEVE request
- 1593
- 1594 • *rs*: The result of the RETRIEVE operation

1595 **8.4 UPDATE**

1596 The UPDATE operation is used to request a partial or complete replacement of the information in
1597 an OIC Resource. The UPDATE operation is initiated by the OIC Client and consists of three steps,
1598 as depicted in Figure 18 and described below.



1599

1600

Figure 18. UPDATE operation

1601 **8.4.1 UPDATE request**

1602 The UPDATE request message is transmitted by the OIC Client to the OIC Server to request the
1603 update of information of an OIC Resource on the OIC Server. The UPDATE request message will
1604 carry the following parameters.

- 1605 • *fr*: Unique identifier of the OIC Client
- 1606 • *to*: URI of the resource targeted for the information update
- 1607 • *ri*: Identifier of the UPDATE request
- 1608 • *op*: UPDATE
- 1609 • *cn*: Information, including properties, of the resource to be updated at the target resource

1610 **8.4.2 Processing by the OIC Server**

1611 Following the receipt of an UPDATE request, the OIC Server may validate if the OIC Client has
1612 the appropriate rights for updating the requested data. If the validation is successful the OIC Server
1613 updates the target resource information according to the information carried in *cn* parameter of the
1614 UPDATE request message. The OIC Server caches the value of *ri* parameter in the UPDATE
1615 request for use in the response.

1616 **8.4.3 UPDATE response**

1617 The OIC Server shall send an UPDATE response message in response to an UPDATE request
1618 message from an OIC Client. The UPDATE response message will include the following
1619 parameters.

- 1620 • *fr*: Unique identifier of the OIC Server
- 1621 • *to*: Unique identifier of the OIC Client
- 1622 • *ri*: Identifier included in the RETRIEVE request
- 1623 • *rs*: The result of the UPDATE operation

1624 **8.5 DELETE**

1625 The DELETE operation is used to request the removal of an OIC Resource. The DELETE operation
1626 is initiated by the OIC Client and consists of three steps, as depicted in Figure 19 and described
1627 below.

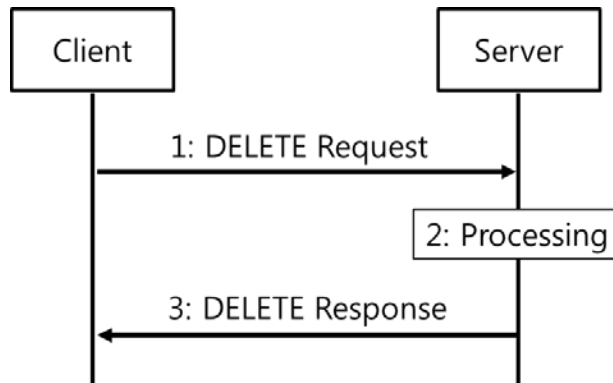


Figure 19. DELETE operation

1628

1629

1630 **8.5.1 DELETE request**

1631 DELETE request message is transmitted by the OIC Client to the OIC Server to delete an OIC
 1632 Resource on the OIC Server. The DELETE request message will carry the following parameters:

- 1633 • *fr*: Unique identifier of the OIC Client
- 1634 • *to*: URI of the target resource which is the target of deletion
- 1635 • *ri*: Identifier of the DELETE request
- 1636 • *op*: DELETE

1637 **8.5.2 Processing by the OIC Server**

1638 Following the receipt of a DELETE request, the OIC Server may validate if the OIC Client has the
 1639 appropriate rights for deleting the identified resource, and whether the identified resource exists.
 1640 If the validation is successful, the OIC Server removes the requested resource and deletes all the
 1641 associated information. The OIC Server caches the value of *ri* parameter in the DELETE request
 1642 for use in the response.

1643 **8.5.3 DELETE response**

1644 The OIC Server shall transmit a DELETE response message in response to a DELETE request
 1645 message from an OIC Client. The DELETE response message will include the following parameters.

- 1646 • *fr*: Unique identifier of the OIC Server
- 1647 • *to*: Unique identifier of the OIC Client
- 1648 • *ri*: Identifier included in the DELETE request
- 1649 • *rs*: The result of the DELETE operation

1650 **8.6 NOTIFY**

1651 The NOTIFY operation is used to request asynchronous notification of state changes. Complete
 1652 description of the NOTIFY operation is provided in section 11.4. The NOTIFY operation uses the
 1653 NOTIFICATION response message which is defined here.

1654 **8.6.1 NOTIFICATION response**

1655 The NOTIFICATION response message is sent by an OIC Server to notify the URLs identified by
 1656 the OIC Client of a state change. The NOTIFICATION response message carries the following
 1657 parameters.

- 1658 • *fr*: Unique identifier of the OIC Server
- 1659 • *to*: URI of the OIC Resource target of the NOTIFICATION message

- 1660 • *ri*: Identifier included in the CREATE request
- 1661 • *op*: NOTIFY
- 1662 • *cn*: The updated state of the resource

1663 **9 Network and connectivity**

1664 **9.1 Introduction**

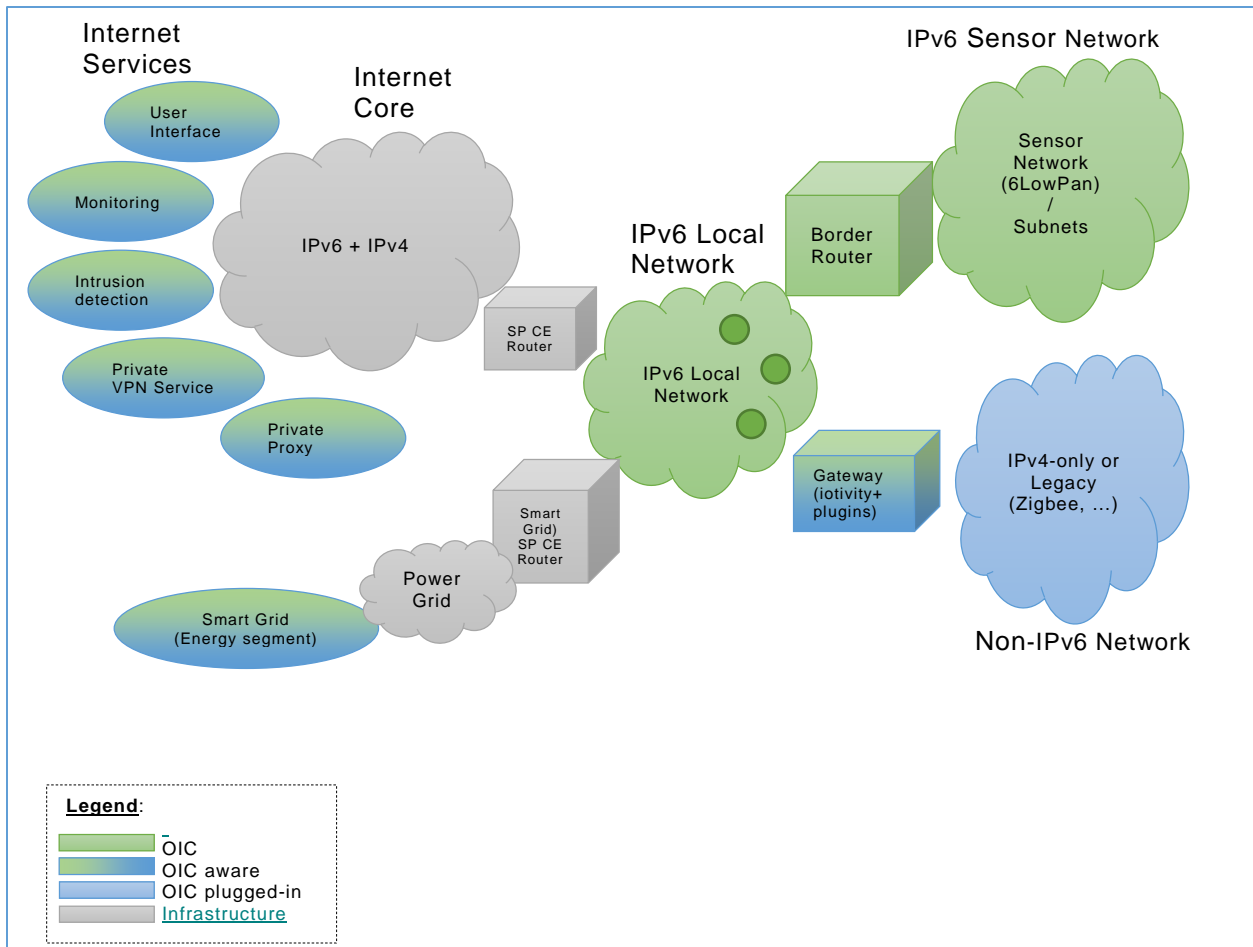
1665 The IOT environment, which the OIC is addressing, is composed of very heterogeneous systems.
1666 Because these systems are often tailored to address dedicated requirements, they are composed
1667 of very diverse products and services. Those products span from very constrained devices that
1668 run on batteries to every day high end devices available on consumer market shelves. The lack of
1669 a global standard and the need to create such a standard has led various groups to work on
1670 streamlining those technologies with well-established networking standards.

1671 The IETF recognized the market transition and realized that Ipv4 was no longer adequate. Not only
1672 does the new scale call for a new technology, but also the manageability of even more diverse
1673 devices, the complexity of multiple subnets and higher security and privacy needs require a whole
1674 new set of standards. Cognizant of the existence and need for dedicated PHY/DLL layers, the IETF
1675 set up working groups to streamline the various existing technologies at the network layer. In
1676 accordance with these market realities, this specification also means to leverage existing radio
1677 silicon (e.g., Bluetooth, Wi-Fi, or 802.15.4) and concentrates on the network layer and the
1678 associated DLL adaptations produced by the IETF.

1679 **9.2 Architecture**

1680 While the aging IPv4 centric network has evolved to support complex topologies, its deployment
1681 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More
1682 complex network topologies, often seen in residential home, are mostly introduced through the
1683 acquisition of additional home network devices, which rely on technologies, like private Network
1684 Address Translation's (NAT's). These technologies require expert assistance to set up correctly
1685 and should be avoided in a home network as they most often result in breakage of constructs like
1686 routing, naming and discovery services.

1687 The multi-segment ecosystem OIC addresses will not only cause a proliferation of new devices
1688 and associated routers, but also new services introducing additional edge routers. All these new
1689 requirements require advance architectural constructs to address complex network topologies like
1690 the one shown in Figure 20.



1691

1692

Figure 20. High Level Network & Connectivity Architecture

1693

Devices depicted in Figure 20 assume one of several roles:

1694

- IPv6 Node, IPv6 Router and IPv6 Host as defined in IETF RFC 6434

1695

- CE Router (Customer Edge Router): As defined in IETF RFC 7084.

1696

- 6LN (6LoWPAN Node), 6LR (6LoWPAN Router), 6LBR (6LoWPAN Border Router) as defined in IETF RFC 6775.

1697

1698

- IPv6 Translator: A device which translates and routes messaging between IPv6 and non-IPv6 networks. An example of a translator is the gateway in Figure 20.

1699

1700

- Constrained Nodes: a node that due to the constrained environment (limited processing power, memory, non-volatile storage and transmission capacity) requires special adaptation layers under the IP Network layer and requires dedicated routing protocols. Examples include devices transmitting over low power PHY's like IEEE 802.14.5, ITU G9959, BLUETOOTH Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),

1701

1702

1703

1704

1705

9.3 IPv6 network layer requirements

1706

9.3.1 Introduction

1707

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

1708

1709

1710

1711 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide
1712 variety of applications such as Web browsing, email, voice, video, and critical system monitoring
1713 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which
1714 is that available address space has been consumed.

1715 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. The OIC
1716 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 1717 • Larger address space. Side-effect: greatly reduce the need for NATs.
- 1718 • More flexible addressing architecture. Multiple addresses and types per interface: Link-local,
1719 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed
1720 networks, better re-numbering capability, etc.
- 1721 • More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 1722 • Technologies enabling IP connectivity on constrained nodes / are based upon IPv6.
- 1723 • All major consumer operating systems (IoS, Android, Windows, Linux) are already IPv6 enabled.
- 1724 • Major Service Providers around the globe are deploying IPv6.

1725 **9.3.2 IPv6 node requirements**

1726 **9.3.2.1 Introduction**

1727 In order to ensure network layer services interoperability from node to node, mandating a common
1728 network layer across all nodes is vital. The protocol should enable the network to be: secure,
1729 manageable, scalable and to include constrained and self-organizing meshed nodes. OIC
1730 recommends IPv6 as the common network layer protocol to ensure interoperability across all OIC
1731 devices. More capable devices may also include additional protocols creating multiple-stack
1732 devices. The remaining of this section will focus on interoperability requirements for IPv6 hosts,
1733 IPv6 constrained hosts and IPv6 routers. The various protocol translation permutations included
1734 in multi-stack gateway devices may be addresses in subsequent addendums of this specification.

1735 **9.3.2.2 IP Layer**

1736 An IPv6 node should support IPv6. If a node supports IPv6, then it shall conform to the
1737 requirements for communication on the local network as follows:

- 1738 • Shall support IETF RFC 2460 “Internet Protocol version 6 Specification” and related updates
1739 as defined in section 5.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1740 • Shall support IETF RFC 4291 “IP Version 6 Addressing Architecture” and related updates as
1741 defined in section 5.9.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1742 • Shall support IETF RFC 4861 “Neighbor Discovery for IPv6” and related updates as defined in
1743 section 5.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1744 • Shall support IETF RFC 4862 “IPv6 Stateless Address Autoconfiguration” and related updates
1745 as defined in section 5.9.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1746 • Shall support IETF RFC 4443 “Internet Control Message Protocol (ICMPv6) for IPv6” [RFC4443]
1747 and related updates as defined in section 5.8 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1748 • Shall support IETF RFC 1981 “Path MTU Discovery” and related updates as defined in section
1749 5.6 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1750 • Shall support IETF RFC 4193 “Unique Local IPv6 Unicast Addresses” and related updates.
- 1751 • Shall support IETF RFC 3810 “Multicast Listener Discovery Version 2 (MLDv2) for IPv6” and
1752 related updates. In particular, shall generate new MLDv2 Report messages for every “All CoAP
1753 Nodes” address FF0X::FD joined on an interface.

1754 **9.3.3 IPv6 router**

1755 An IPv6 router shall support all node requirements defined in section 9.3.2.

1756 **9.3.4 IPv6 host**

1757 An Ipv6 host shall support all node requirements defined in section 9.3.2.

1758 **9.3.5 IPv6 constrained nodes**

1759 **9.3.5.1 Requirements**

1760 An IPv6 constrained node shall support all node requirements defined in section 9.3.2. If a
1761 constrained node supports IPv6, it should use the adaptations defined in sections in order to
1762 support IPv6.

1763 **9.3.5.2 IP layer**

1764 An IPv6 constrained node should support the neighbour discovery optimization as defined in
1765 IETF RFC 6775 "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal
1766 Area Networks (6LoWPANs)".

1767 **9.3.5.3 Sub IP layer**

- 1768 • An IPv6 constrained node on an ITU-T G.9959 network should support IETF RFC 7428 and
1769 related updates.
- 1770 • An IPv6 constrained node on an IEEE 802.15.4 network should support IETF RFC 4944 and
1771 related updates.
- 1772 • An IPv6 constrained node on a BLUETOOTH(R) Low Energy network should support
1773 IETF draft-ietf-6lo-btle-14 and related updates.

1774 **10 Endpoint discovery**

1775 **10.1 Introduction**

1776 This section describes how an OIC Endpoint is discovered by another OIC Endpoint in a network.
1777 An OIC Endpoint shall support CoAP discovery. HTTP discovery is optional for an OIC Endpoint.
1778 For CoAP transport protocol the Endpoint discovery uses mechanisms as defined in
1779 IETF RFC 7252. If HTTP discovery is supported, then Endpoint discovery uses mDNS as defined
1780 in IETF RFC 6762. Endpoint discovery is a Multicast discovery which is not required when the
1781 Endpoint is already known.

1782 **10.2 CoAP based Endpoint discovery**

1783 The following is the description of CoAP based Endpoint discovery:

- 1784 a) All advertising or publishing OIC Devices shall join the 'All CoAP Nodes' multicast group, i.e.
1785 FF0X::FD for IPv6 and listen on the port "5683".
- 1786 b) The OIC Client that wants to discover resources shall first join the 'All CoAP Nodes' multicast
1787 group.
- 1788 c) Then the OIC Client shall send a discovery request (GET request) to the multicast group 'All
1789 CoAP Nodes' and port 5683 where the URI in the request shall be /oic/res.
- 1790 d) If the OIC Client is in the process of discovering a particular resource type, then it shall use
1791 the Query mechanism (section 6.2.1) with key "rt" and the wanted target as value of rt
- 1792 e) When "rt" Query key is omitted all OIC Devices shall respond to that request
- 1793 f) The considerations for handling multicast requests shall be as described in section 8 of
1794 IETF RFC 7252 and section 4.1 in IETF RFC 6690.

1795 g) The OIC Devices that receive this request shall respond using CBOR as the payload (content)
1796 encoding. An OIC Device shall indicate support for CBOR as an additional supported payload
1797 (content) encoding for multicast discovery as described in section 12.2.3. An OIC Device shall
1798 respond to a received multicast discovery that indicates support for CBOR using CBOR as the
1799 payload (content) encoding. In later versions of the specification other formats could be
1800 included (e.g., JSON, XML/EXI).

1801

1802 Below are a few examples to search for OIC Devices on the network:

1803 To search for all OIC Devices on the network an OIC Client can issue:

1804 Request

1805 GET /oic/res

1806 Response

```
1807 [
1808   {
1809     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1810     "links": [
1811       {
1812         "href": "/door",
1813         "rt": "oic.r.door",
1814         "if": "oic.if.b oic.ll"
1815       },
1816       {
1817         "href": "/door/lock",
1818         "rt": "oic.r.lock",
1819         "if": "oic.if.b",
1820         "type": "application/cbor application/exi+xml"
1821       }
1822     ]
1823   },
1824   {
1825     "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
1826     "links": [
1827       {
1828         "href": "/light",
1829         "rt": "oic.r.light",
1830         "if": "oic.if.s"
1831       },
1832       {
1833         "href": "/binarySwitch",
1834         "rt": "oic.r.switch.binary",
1835         "if": "oic.if.a",
1836         "type": "application/cbor"
1837       }
1838     ]
1839   }
1840 ]
```

1841 To search for oic.r.light resources on the network an OIC Client can issue:

1842 Request

1843 GET /oic/res?rt="oic.r.light"

1844 Response

```
1845 [
1846   {
1847     "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
1848     "links": [
1849       {
1850         "href": "/light",
1851         "rt": "oic.r.light",
1852         "if": "oic.if.s"
1853       }
1854     ]
1855   }
1856 ]
```

1855 }
 1856]
 1857

1858 To search for oic.r.switch.binary resources on the network an OIC Client can issue:

1859 **Request**

1860 GET /oic/res?rt="oic.r.switch.binary"

1861 **Response**

```
1862 [
1863   {
1864     "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9",
1865     "links": [
1866       {
1867         "href": "/binarySwitch",
1868         "rt": "oic.r.switch.binary",
1869         "if": "oic.if.a",
1870         "type": "application/cbor"
1871       }
1872     ]
1873   }
1874 ]
```

1875

1876 Note that the examples do not indicate the multicast address and port number. The examples also do not include the
 1877 accept header.

1878

1879 **11 Functional interactions**

1880 **11.1 Introduction**

1881 The functional interactions between an OIC Client and an OIC Server are described in section 11.2
 1882 through section 11.6 respectively. The functional interactions use CRUDN messages (section 8)
 1883 and include Discovery, Notification, and Device management. These functions require support of
 1884 core defined resources as defined in Table 12. More details about these resources are provided
 1885 later in this section.

1886

Table 12. List of OIC Core Resources

Fixed URI	Resource Type Title	Related Functional Interaction	Requirement (M/CR/O)
/oic/res	Default	Discovery	M
/oic/p	Platform	Discovery	M
/oic/d	Device	Discovery	M
/oic/rts	Resource Type	Discovery	CR
/oic/ifs	Interface	Discovery	CR
/oic/con	Configuration	Device Management	CR
/oic/mon	Monitoring	Device Management	CR
/oic/mnt	Maintenance	Device Management	CR

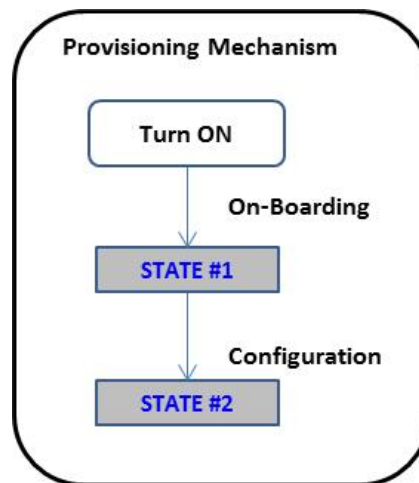
1887

1888 11.2 Provisioning

1889 Provisioning in OIC Framework includes two distinct processes: On-Boarding and Configuration.

1890 On-Boarding is the process which delivers required information to an OIC Device for joining the
1891 OIC network. When On-Boarding process is completed, the OIC Device has necessary information
1892 and is able to join the OIC network (State #1 in Figure 21). Further details about provisioning can
1893 be found in OIC Security specification (Owner PSK).

1894 Configuration is the process which delivers required information to a device for accessing OIC
1895 services. At the end of the configuration process, the OIC Device has all the necessary information
1896 and is able to access OIC services (State #2 in Figure 21).



1897

1898 **Figure 21. Provisioning State Changes**

1899 #1 On-Boarding

1900 OIC Framework is applicable to many different types of devices with different capabilities, including
1901 devices with a rich user interface that can take inputs from the users, e.g., smartphones, as well
1902 as headless devices that have no means for receiving user inputs, e.g., presence sensors.
1903 Additionally, the OIC Devices may support different communication and connectivity technologies,
1904 e.g., Bluetooth, Wi-Fi, etc. Different communication and connectivity technologies provide different
1905 on-boarding mechanisms specific to that technology.

1906 Due to these differences and diversity of device capabilities, this version of specification does not
1907 mandate a particular process for On-Boarding, instead, specifies the state of the OIC Device upon
1908 completion of the OnBoarding process.

1909 As part of the On-Boarding process the device acquires detailed information and required
1910 parameter values to be able to connect to the network, resulting in successful establishment of a
1911 connection to the network at the end of the On-Boarding process. The required information and
1912 parameters values include for example, SSID for Wi-Fi as well as authentication credentials.

1913 Later versions of this specification may specify a common process for On-Barding across different
1914 communication and connectivity technologies.

1915 #2 Configuration

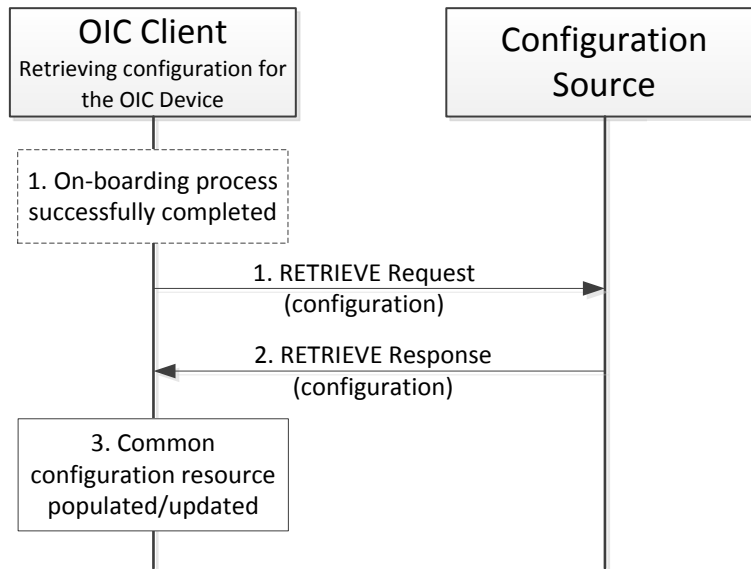
1916 Once an OIC Device is successfully connected to the OIC network, it needs additional configuration
1917 information for accessing the OIC services or to subscribe for OIC services. The information
1918 required may include geographical location, time zone, security requirements, etc. This information
1919 may be pre-loaded on an OIC Device, or may be acquired from a configuration service that can
1920 be located on another OIC Device, e.g., the Configuration Source. The information regarding the
1921 configuration service resource, e.g., the URI of the Configuration Source, is pre-configured on the
1922 OIC Device.

1923 The configuration information is also in OIC core resource '/oic/con'. Upon completion of the On-
1924 Boarding process and as soon as the OIC Device is connected to the network, if the configuration
1925 information is not pre-loaded, it shall initiate the configuration process, as a result of which the
1926 OIC Device acquires the relevant configuration information, through either a pull or a push
1927 interaction, and populates its designated configuration resource with its current configured state
1928 information. The designated configuration resource maintains the latest configuration state and is
1929 the designated resource through which updates to the configuration are made.

1930 If the configuration information is not pre-loaded the OIC Device retrieves them from the
1931 Configuration Source. During the lifetime of an OIC Device an OIC Client may retrieve or update
1932 the configuration state of the OIC Device. Some of the configuration information is read only and
1933 some may be modified by Configuration Sources depending on the 'Access Modes' of properties
1934 in /oic/con resource.

1935 Figure 22 depicts the interactions triggered by an OIC Device to retrieve its configuration
1936 information from the Configuration Source which may be located on a remote OIC Device or locally.
1937 These interactions occur instantly following completion of On-Boarding process; the OIC Device
1938 may retrieve its configuration at any time during its lifetime.

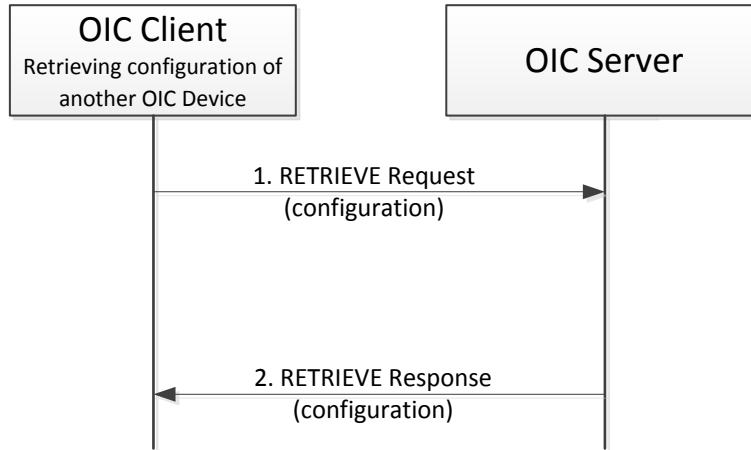
1939



1940

1941 **Figure 22. Interactions initiated by the OIC Device to retrieve its configuration from a**
1942 **configuration source**

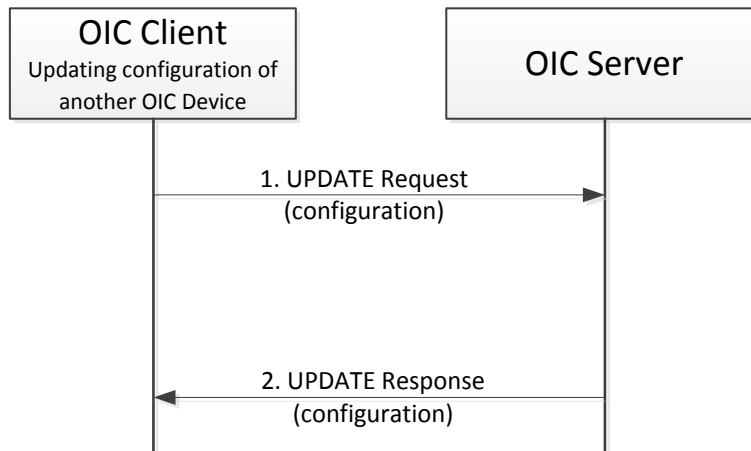
1943 Figure 23 depicts the interactions when the retrieve of configuration information is done by an OIC
1944 Client.



1945
1946

Figure 23. Interactions for retrieving the configuration state of an OIC Device.

1947 Figure 24 depicts the interactions when the configuration information of an OIC Device is updated
1948 by an OIC Client, e.g., the Configuration Source.
1949



1950
1951

Figure 24. Update of and OIC Device configuration

1952 If Configuration is supported by an OIC Device, i.e., the configuration information may be
1953 dynamically updated, the OIC Core Resource /oic/con shall be supported as the designated
1954 configuration resource as described in Table 15.

1955 **Configuration Resource**

1956 An OIC Device or an OIC Platform may be initially configured from information that is set or
1957 provisioned at bootstrap. In addition, the OIC Device and OIC Platform may be configured further
1958 by an external agent post bootstrap depending on changing conditions or context. The core
1959 resource /oic/con exposes properties that may be used to effect changes in the configuration.
1960

1961 A configuration is determined by setting all the properties that collectively pertain to that
1962 configuration. The outcome of setting a new configuration is determined by the value of the specific
1963 properties in that set. Setting a new configuration through /oic/con may lead to initiation of
1964 processes that affect or create side effects in other resources.
1965

1966

Table 13. Configuration Resources

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/con	Configuration	oic.wk.con	oic.if.rw	The resource through which configurable information specific to the OIC Device is exposed. The resource properties exposed by /oic/con are listed in Table 14.	Configuration

1967

1968 Table 14 defines the oic.wk.con resource type.

1969

1970

Table 14. oic.wk.con resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R, W	yes	Human friendly name For example, "Bob's Thermostat"
Location	loc	json (has two attributes one with longitude and latitude and also a name for a location)			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 (").

1971

11.3 Resource discovery**11.3.1 Introduction**

1974 Discovery is a function which enables endpoint discovery as well as resource based discovery.
1975 Endpoint discovery is described in detail in section 10. This section mainly describes the resource
1976 based discovery.

11.3.2 Resource based discovery: mechanisms**11.3.2.1 Overview**

1979 As part of discovery, an OIC Client may find appropriate information about other OIC peers. This
1980 information could be instances of resources, resource types or any other information represented
1981 in the resource model that an OIC peer would want another OIC peer to discover.

1982 At the minimum, Resource based discovery uses the following:

- 1983 1) A resource to enable discovery shall be defined. The representation of that resource shall
1984 contain the information that can be discovered.
- 1985 2) The resource to enable discovery shall be a) specified and commonly known a-priori or at
1986 bootstrap (e.g., specified in vertical specification) or b) discovered (e.g., using out of band
1987 methods)
- 1988 3) An OIC Device for hosting the resource to enable discovery shall be identified.
- 1989 4) A mechanism and process to publish the information that needs to be discovered with the
1990 resource to enable discovery.
- 1991 5) A mechanism and process to access and obtain the information from the resource to enable
1992 discovery. A query may be used in the request to limit the returned information.
- 1993 6) A scope for the publication
- 1994 7) A scope for the access.
- 1995 8) A policy for visibility of the information.

1996

1997 Depending on the choice of the base aspects defined above, the OIC Framework defines three
1998 resource based discovery mechanisms:

- 1999 • Direct discovery, where the OIC Resources are published locally at the OIC Device
2000 hosting the resources and are discovered through peer inquiry.
- 2001 • Indirect discovery, where OIC Resources are published at a third party assisting with the
2002 discovery and peers publish and perform discovery against the resource to enable
2003 discovery on the assisting 3rd party.
- 2004 • Presence, where the resource to enable discovery is hosted local to the initiator of the
2005 discovery inquiry but remote to the OIC Devices that are publishing discovery
2006 information.

2007 An OIC Device shall support direct discovery.

2008 **11.3.2.2 Direct discovery**

2009 In direct discovery,

- 2010 1) The OIC Device that is providing the information shall host the resource to enable
2011 discovery.
 - 2012 2) The OIC Device publishes the information available for discovery with the local resource
2013 to enable discovery (i.e. local scope).
 - 2014 3) OIC Clients interested in discovering information about this OIC Device shall issue
2015 RETRIEVE requests directly to the resource. The request may be made as a unicast or
2016 multicast. The request may be generic or may be qualified or limited by using appropriate
2017 queries in the request.
 - 2018 4) The “server” OIC Device that receives the request shall send a response with the
2019 discovered information directly back to the requesting “client” OIC Device.
 - 2020 5) The information that is included in the request is determined by the policies set for the
2021 resource to be discovered locally on the responding OIC Device.
- 2022

2023 **11.3.2.3 Indirect discovery of OIC Resources (resource directory based discovery)**

2024 In indirect discovery the information about the resource to be discovered is hosted on an OIC
2025 Server that is not hosting the resource. See section 11.3.6 for details on resource directory based
2026 discovery.

2027 In indirect discovery:

- 2028 a) The resource to be discovered is hosted on an OIC Device that is neither the client
2029 initiating the discovery nor the OIC Device that is providing or publishing the information
2030 to be discovered. This OIC Device may use the same resource to provide discovery for
2031 multiple agents looking to discover and for multiple agents with information to be
2032 discovered.
- 2033 b) The OIC Device to be discovered or with information to discover, publishes that
2034 information with resource to be discovered on a different OIC Device. The policies on the
2035 information shared including the lifetime/validity are specified by the publishing OIC
2036 Device. The publishing OIC Device may modify these policies as required.
- 2037 c) The client doing the discovery may send a unicast discovery request to the OIC Device
2038 hosting the discovery information or send a multicast request that shall be monitored and
2039 responded to by the OIC Device. In both cases, the OIC Device hosting the discovery
2040 information is acting on behalf of the publishing OIC Device.
- 2041 d) The discovery policies may be set by the OIC Device hosting the discovery information or
2042 by the party that is publishing the information to be discovered. The discovery information
2043 that is returned in the discovery response shall adhere to the policies that are in effect at
2044 the time of the request.

2045

2046 **11.3.2.4 Advertisement Discovery**

2047 In advertisement discovery:

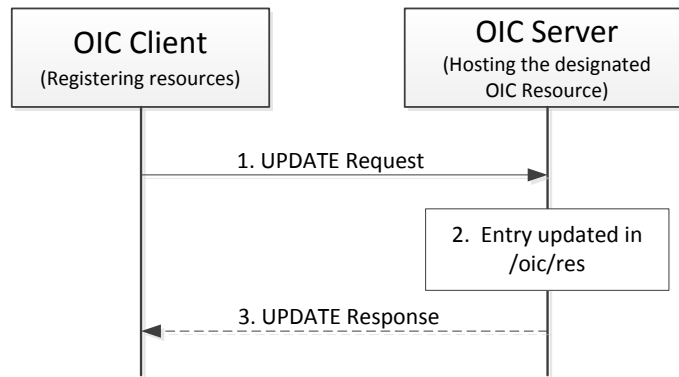
- 2048 a) The resource to enable discovery is hosted local to the OIC Device that is initiating the
2049 discovery request (client). The resource to enable discovery may be an OIC Core Resource
2050 or discovered as part of a bootstrap.
- 2051 b) The request could be an implementation dependent lookup or be a local RETRIEVE request
2052 against the resource that enables discovery.
- 2053 c) The OIC Device with information to be discovered shall publish the appropriate information
2054 to the resource that enables discovery.
- 2055 d) The publishing OIC Device is responsible for the published information. The publishing OIC
2056 Device may UPDATE the information at the resource to enable discovery based on its needs
2057 by sending additional publication requests. The policies on the information that is
2058 discovered including lifetime is determined by the publishing OIC Device.

2059

2060 **11.3.3 Resource based discovery: Information publication process**

2061 The mechanism to publish information with the resource to enable discovery can be done either
2062 locally or remotely. The publication process is depicted in Figure 25. The OIC Device which has
2063 discovery information to publish shall a) either update the resource that enables discovery if
2064 hosted locally or b) issue an UPDATE request with the information to the OIC Device which hosts
2065 the resource that enables discovery. The OIC Device hosting the resource to enable discovery
2066 adds/updates the resource to enable discovery with the provided information and then responds
2067 to the OIC Device which has requested the publication of the resource with an UPDATE response.

2068



2069

2070

Figure 25. Resource based discovery: Information publication process

2071

11.3.4 Resource based discovery: Finding information

2072

2073

2074

2075

2076

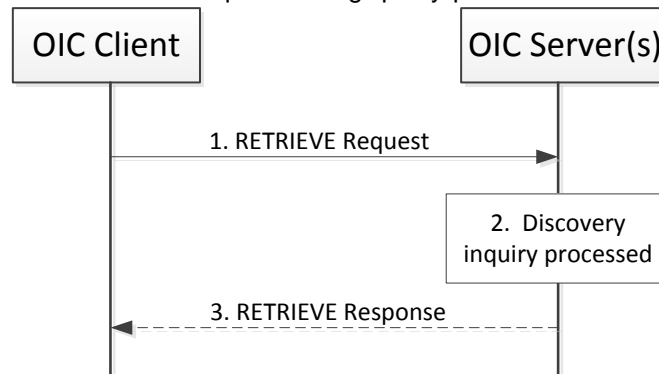
2077

2078

2079

2080

The discovery process (Figure 26) is initiated as a RETRIEVE request to the resource to enable discovery. The request may be sent to a single OIC Device (as in a Unicast) or to multiple OIC Devices (as in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the support in the data connectivity layer. The response to the request has the information to be discovered based on the policies for that information. The policies can determine which information is shared, when and to which requesting agent. The information that can be discovered can be resources, types, configuration and many other standards or custom aspects depending on the request to appropriate resource and the form of request. Optionally the requester may narrow the information to be returned in the request using query parameters in the URI query.



2081

2082

Figure 26. Resource based discovery: Finding information

2083

2084

Discovery Resources

2085

2086

Some of the OIC Core Resources (section 7.2.2) shall be implemented on all OIC Devices to support discovery. The OIC Core Resources that shall be implemented to support discovery are:

2087

- '/oic/res' for discovery of resources

2088

- '/oic/p' for discovery of platform

2089

- '/oic/d' for discovery of device information

2090

Details for these mandatory OIC Core Resources are described in Table 15

2091

Platform resource –

2092 The OIC recognizes that more than one instance of OIC Device may be hosted on a single platform.
 2093 Clients need a way to discover and access the information on the platform. The core resource,
 2094 /oic/p exposes platform specific properties. All instances of OIC Device on the same OIC Platform
 2095 shall have the same values of any properties exposed (i.e. an OIC Device may choose to expose
 2096 optional properties within /oic/p but when exposed the value of that property should be the same
 2097 as the value of that property on all other OIC Devices on that OIC Platform)

2098
 2099 **Device resource**

2100 The device resource shall have the fixed URI /oic/d. The resource /oic/d exposes the properties
 2101 pertaining to an OIC Device as defined in Table 15. The properties exposed are determined by the
 2102 specific instance of OIC Device and defined by the resource type(s) of /oic/d on that OIC Device.
 2103 Since all the resource types of /oic/d are not known a priori, the resource type(s) of /oic/d shall be
 2104 determined by discovery through the core resource /oic/res. The device resource /oic/d shall have
 2105 a default resource type that helps in bootstrapping the interactions with this device (the default
 2106 type is described in Table 15.)

2107
 2108 **Protocol indication**

2109 An OIC Device may need to support different messaging protocols depending on requirements for
 2110 different application profiles. For example, the Smart Home profile may use CoAP and the
 2111 Industrial profile may use DDS. To enable interoperability, an OIC Device uses the protocol
 2112 indication to indicate the transport protocols they support and can communicate over.

2113

2114

Table 15. Mandatory discovery OIC Core Resources

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/res	Default	oic.wk.res	oic.if.ll	The resource through which the corresponding OIC Server is discovered and introspected for available resources. The semantics for /oic/res shall be same as a collection that supports the link-list interface and whose referenced resources is the list of resources that are discoverable on that OIC Server. The resource properties exposed by /oic/res are listed in Table 16.	Discovery
/oic/p	Platform	oic.wk.p	oic.if.r	The resource through which platform specific information is discovered. The resource properties exposed by /oic/p are listed in Table 18	Discovery
/oic/d	Device	oic.wk.d and/or one or more Device Specific resource type IDs	oic.if.r	The discoverable via /oic/res resource which exposes properties specific to the OIC Device instance. The resource properties exposed by /oic/d are listed in Table 18 /oic/d may have one or more resource types that are specific to OIC Device in addition to the default resource type or if present overriding the default resource type. The base type oic.wk.d defines the properties that shall be exposed by all OIC Devices. The device specific resource type(s) exposed are dependent on the class of device (e.g. air conditioner, smoke alarm); applicable values are defined by the vertical specifications.	Discovery

2115

2116 Table 16 defines oic.wk.res resource type.

2117 **Table 16. oic.wk.res resource type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Device Identifier	di	UUID			R	yes	The device identifier as indicated by the /oic/d resource of the device
Links	json	string	See 7.1.6.2		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 17 For example, "1 and 3" indicates that the OIC device supports coap and http as messaging protocols.

2118 An OIC Device which wants to indicate its messaging protocol capabilities may add the property
 2119 'mpro' in response to a request on /oic/res. An OIC Device shall support CoAP based discovery as
 2120 the baseline discovery mechanism (see section 10.2). An OIC Client which sees this property in a
 2121 discovery response can choose any of the supported messaging protocols for communicating with
 2122 the OIC Server for further messages. For example, if an OIC Device supporting multiple protocols
 2123 indicates it supports a value of "1 3" for the 'mpro' property in the discovery response, then it
 2124 cannot be assumed that there is an implied ordering or priority. But a vertical service specification
 2125 may choose to specify an implied ordering or priority. If the 'mpro' property is not present in the
 2126 response, An OIC Client shall use the default messaging protocol as specified in the vertical
 2127 specification for further communication. Table 17 provides an OIC registry for protocol schemes.

2128 **Table 17. Protocol scheme registry**

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

2129 Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by an OIC Client
 2130 to form requests in a different messaging protocol other than discovery is out of scope.

2131
 2132 The following applies to the use of /oic/d as defined above:

- 2133 • /oic/d is a well-known URI for the Device resource
- 2134 • The Device resource has a base resource type ID of oic.wk.d
- 2135 • The base resource type oic.wk.d shall have mandatory properties as defined in Table 3

- 2136 • A vertical may choose to expose its Device Class (e.g., refrigerator or A/C) by overriding the
2137 Resource Type ID set associated with /oic/d.
 - 2138 ○ For example; rt of /oic/d becomes 'oic.d.<thing>'
- 2139 • A vertical may also choose to expose its Device Class (e.g., refrigerator or A/C) by adding to
2140 the Resource Type ID set associated with /oic/d.
 - 2141 ○ For example; rt of /oic/d becomes 'oic.wk.d,oic.d.<thing>'
- 2142 • The properties exposed for the Device Class specified resource type ID are by default the
2143 mandatory properties in Table 3
- 2144 • A vertical may choose to extend the properties exposed via /oic/d with the use of a Device
2145 Class specific resource type ID.
 - 2146 ○ However the mandatory properties defined in Table 3 shall always be present
- 2147 • /oic/d is exposed in /oic/res as an entry in the set of web-links
- 2148 • Should there be more than one resource type ID listed; then the default resource type ID for
2149 /oic/d is the first resource type ID listed. So a vertical can list 'oic.d.thing' first. This then
2150 means a GET /oic/d returns the properties for oic.d.thing.
- 2151 • The mandatory properties need to be in the device specific class type only if that device class
2152 type can be made a default. If a vertical defines more than one Device class resource type then
2153 all of them don't need to carry these base mandatory properties.

2154 Note:

2155 As per existing Core specification definitions the resource type ID may be a list of resource type IDs; when that is the
2156 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.
2157 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
2158 for the rt listed in the query.

2159 Table 18 oic.wk.d resource type definition defines the base resource type for the /oic/d resource.

2160

2161

Table 18. oic.wk.d resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R	yes	Human friendly name For example, "Bob's Thermostat"
Spec Version	lcv	string			R	yes	Spec version of the core specification this device is implemented to, The syntax is "core.major.minor"]
Device ID	di	UUID			R	yes	Unique identifier for OIC Device (UUID)
Data Model Version	dmv	CSV			R	yes	Spec version(s) of the vertical specifications this device data model is implemented to. The syntax is a comma separated list of "<vertical>.major.minor"]. <vertical> is the name of the vertical (i.e. sh for Smart Home)

2162

2163 The additional resource type(s) of the /oic/d resource are defined by the vertical specification.

2164

2165 Table 18 defines oic.wk.p resource type.

Table 19. 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	Platform Identifier
Manufacturer Name	mnmn	string			R	yes	Name of manufacturer (not to exceed 16 characters)
Manufacturer Details Link (URL)	mnml	URL			R	no	URL to manufacturer (not to exceed 32 characters)
Model Number	mnmo	string			R	no	Model number as designated by manufacturer
Date of Manufacture	mndt	date		Time (<i>show RFC</i>)	R	no	Manufacturing date of device
Platform Version	mpv	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 device firmware
Support URL	mnsi	URL			R	no	URL that points to support information from manufacturer
SystemTime	st	datetime			R	no	Reference time for the device

2169 Composite Device

2170 A physical device may be modelled as a single device or as a composition of other devices. For
 2171 example a refrigerator may be modelled as a composition, as such part of its definition of may
 2172 include a sub-tending thermostat device which itself may be composed of a sub-tending
 2173 thermometer device.

2174 There may be more than one way to model an OIC server as a composition. One example method
 2175 would be to have OIC Platform which represents the composite device to have more than one
 2176 instance of an OIC Device on the OIC Platform. Each OIC Device instance represents one of the
 2177 distinct devices in the composition. Each instance of OIC Device may itself have or host multiple
 2178 instances of other resources.

2179 An implementation irrespective of how it is composed shall only expose a single instance of /oic/d
 2180 with an 'rt' of choice for each logical OIC Server.

2181 Thus, for the above refrigerator example if modeled as a single OIC Server; /oic/res would expose
 2182 /oic/d with a resource type name appropriate to a refrigerator. The sub-tending thermostat and

2183 thermometer devices would be exposed simply as instances of a resource with a device
 2184 appropriate resource type with an associated URI assigned by the implementation; e.g.,
 2185 /MyHost/MyRefrigerator/Thermostat and /MyHost/MyRefrigerator/Thermostat/Thermometer.

2186

2187 **Additional Discovery Resources**

2188 The OIC Core Resources that may be implemented to support additional discovery are:

- 2189 ● ‘/oic/rts’ for discovery of resource types
- 2190 ● ‘/oic/ifs’ for discovery of interfaces
- 2191 ● ‘/oic/ad’ for presence discovery

2192 Details for these optional OIC Core Resources are described in Table 20

2193 **Table 20. Optional discovery OIC Core Resources**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/rts	Resource Type	oic.wk.rts	oic.if.r	The resource through which the supported resource types on an OIC Device are identified. The resource types can be pre-loaded or pre-configured at build time or may be downloaded at run time. This resource is read-only which implies that resource types cannot be pushed to the OIC Device (this does not preclude the OIC Device from "downloading" new types from a resource type registry/directory and then exposing those as supported). Resource types discovered through /oic/rts is used for specifying the resource types for resources that are CREATED. This resource is modelled as a simple resource. The resource properties exposed by /oic/rts are listed in Table 21.	Discovery
/oic/ifs	Interface	oic.wk.ifs	oic.if.r	The resource through which the supported interfaces are identified. This is interface information per-resource returned as part of resource discovery. This resource announces the supported interfaces that may be used for creating resources on this OIC Server. The resource properties exposed by /oic/ifs are listed in Table 22.	Discovery
/oic/ad	Advertisement	oic.wk.ad	oic.if.rw	The resource through which the OIC Devices advertise their presence. This allows "advertisement" to be enabled as a mode of discovery at the resource model level. Advertisements could be for resources, presence, state changes, etc. The advertisement is done as a CREATE or UPDATE request to this resource. This resource is modelled as a collection. The resource type of resources i.e. oic.wk.ad exposed by /oic/ad are listed in Table 23.	Discovery

2194
 2195 Table 21 defines oic.wk.rts resource type.
 2196

2197

Table 21. oic.wk.rts supported resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Type list	tl	BSV	0 to many resource type IDs		R	yes	List of the resource types supported by the OIC Device.

2198

2199 Table 22 defines oic.wk.ifs resource type.

2200

2201

Table 22. oic.wk.ifs resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Interface list	il	BSV	0 to many resource interfaces		R	yes	List of the resource interfaces supported by the OIC Device.

2202

2203 Table 23 defines oic.wk.ad resource type. This is the type of the resource that is posted (advertised)
 2204 to the /oic/ad resource to announce the presence of a resource. The resource /oic/ad is a collection
 2205 with each resource of type oic.wk.ad. To announce the presence of a device the fully qualified
 2206 resource URI (from /oic/d) is published along with resource type as oid.wk.d.
 2207

2207

2208

Table 23. oic.wk.ad resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Time to Live	ttl	integer	Seconds		R, W	Yes	Time to live for this information expressed in seconds.
Nonce	non	integer	A random number or monotonically increasing from 1. If it is just monotonically increasing it will roll over causing duplicates over some period of time.		R, W	Yes	The nonce is used to make sure that the current representation is different from another representation that the resource may have been updated with before. This allows OIC Clients that monitor this to know when the different representation is available or if they do an RETRIVE

							operation on the resource then they can compare with a previous RETRIEVE operation to make sure this is a new representation.
Announcement Trigger	trg	enum	create, change, delete		R, W	Yes	Reason for this announcement can be creation or deletion of the resource or change, to one or more properties in a resource.
Resource URI	href	URI			R	No	Fully qualified URI of the resource whose presence is being announced.
Resource type	rt	BSV	1 to many (as space separated)		R	Yes	Resource type

2209

2210 11.3.5 Resource discovery using '/oic/res'

2211 Discovery using '/oic/res' is the default discovery mechanism that shall be supported by all OIC
2212 Devices as follows:

- 2213 a) Every OIC Device updates its local '/oic/res' with the resources that are discoverable (see
2214 section 7.1.4.2.4). Every time a new resource is instantiated on the OIC Device and if that
2215 resource is discoverable by a remote OIC Device then that resource is published with the
2216 '/oic/res' resource that is local to the OIC Device (as the instantiated resource).
- 2217 b) An OIC Device wanting to discover resources or resource types on one or more remote OIC
2218 Devices makes a RETRIEVE request to the /oic/res on the remote OIC Devices. This request
2219 may be sent multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE
2220 request may optionally be restricted using appropriate clauses in the query portion of the
2221 request. Queries may select based on resource types, interfaces, or properties.
- 2222 c) Query applies to the representation of the resources. '/oic/res' is the only resource whose
2223 representation has "rt". So '/oic/res' is the only resource that can be used for Multicast
2224 discovery at the transport protocol layer.
- 2225 d) The OIC Device receiving the RETRIEVE request responds with a list of resources, the
2226 resource type of each of the resources and the interfaces that each resource supports.
2227 Additionally information on the policies active on the resource can also be sent. The policy
2228 supported includes observability and discoverability. (More details below)
- 2229 e) The receiving OIC Device may do a deeper discovery based on the resources returned in the
2230 request to /oic/res.

2231

2232 The information that is returned on discovery against /oic/res is at the minimum:

- 2233 • The URI (specifically fully qualified URL) of the resource

- 2234 • The Resource Type of each resource. More than one Resource Type may be returned if the
2235 resource enables more than one type. To access resources of multiple types, the specific
2236 resource type that is targeted shall be specified in the request.
- 2237 • The OIC Interfaces supported by that resource. Multiple interfaces may be returned. To access
2238 a specific interface that interface shall be specified in the request. If the interface is not
2239 specified, then the Default Interface is assumed.
- 2240 • Policies defined against that resource. These policies may be security related, access modes,
2241 types of interactions, etc. In addition to the request/response type of interaction, the
2242 specification allows the resource to be “observed” (section 11.4.2).

2243

2244 The JSON schemas for discovery using ‘/oic/res’ are described in Figure 7 and Figure 9 in section
2245 7.1.6.2. Also refer to Section 10 (Endpoint Discovery) for details of Multicast discovery using
2246 /oic/res on a CoAP transport.

2247 After performing discovery using /oic/res, OIC Clients may discover additional details about OIC
2248 Server by performing discovery using /oic/p, /oic/rts etc. If an OIC Client already knows about OIC
2249 Server it may discover using other resources without going through the discovery of /oic/res.

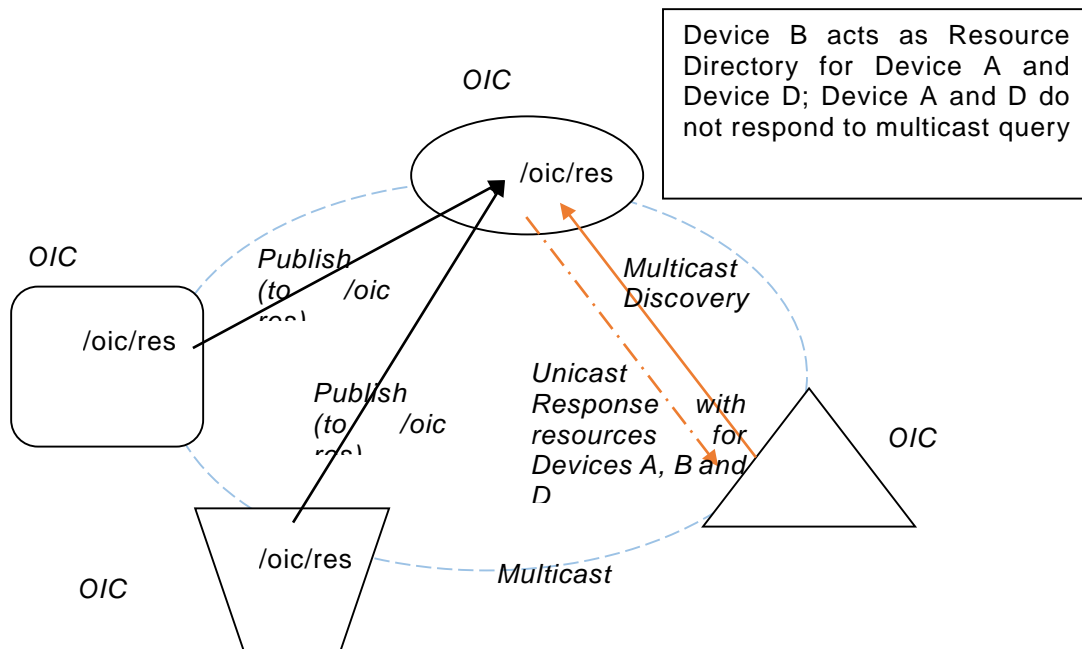
2250 **11.3.6 Resource directory (RD) based discovery**

2251 **11.3.6.1 Introduction**

2252 **11.3.6.1.1 Indirect discovery for lookup of the resources**

2253 Direct discovery is the mechanism used currently to find resources in the network. When needed,
2254 resources are queried at a particular node directly or a multicast packet is sent to all nodes. Each
2255 queried node responds directly with its discoverable resources to the discovering device.
2256 Resources available locally are registered on the same device.

2257 In some situations, one of the other mechanisms described in section 11.3.2.3, called indirect
2258 discovery, may be required. Indirect discovery is when a 3rd party device, other than the
2259 discovering device and the discovered device, assists with the discovery process. The 3rd party
2260 only provides information on resources on behalf of another device but does not host resources
2261 on part of that device.



2262
2263 **Figure 27. Indirect discovery of resource by resource directory**

2264 Indirect discovery is useful for a resource constrained device that needs to sleep to manage power
2265 and cannot process every discovery request, or when devices may not be on the same network
2266 and requires optimization for discovery. Once resources are discovered using indirect discovery
2267 then the access to the resource is done by a request directly to the OIC Device that hosts that
2268 resource.

2269 **11.3.6.1.2 Resource directory**

2270 A resource directory (RD) is an OIC Device that assists with indirect discovery. A RD can be
2271 queried at its /oic/res resource to find resources hosted on other OIC Devices. These OIC Devices
2272 can be sleepy nodes or any other device that cannot or may not respond to discovery requests.
2273 OIC Device can publish all or partial list of resources they host to a RD. The RD then responds to
2274 queries for resource discovery on behalf of the publishing OIC Device (for example: when an OIC
2275 Device may go to sleep). For general resource discovery, the RD behaves like any other OIC
2276 Server in responding to requests to /oic/res.

2277 Any OIC Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The OIC
2278 Devices that want to discover RDs shall use this resource and one of the resource discovery
2279 mechanisms to discover the RD and get the parameters of the RD. The information discovered
2280 through this resource shall be used to select the appropriate RD to use for resource publication.
2281 The weighting information shall include the following criteria: power source (AC, battery powered
2282 or safe/reliable), connectivity (wireless, wired), CPU, memory, load statistics (processing
2283 publishing and query from the devices). In addition, the RD shall return a bias factor that ranges
2284 from 0 to 100. Optionally, the RD may also return a context - the value which shall be a string and
2285 semantics of the context are not discussed in this document but it is expected that the context will
2286 be used to establish a domain, region or some such scope that is meaningful to the application,
2287 deployment or usage.

2288 Using these criteria or the bias factor, the OIC Device shall select one RD (per context) to publish
2289 its resources. A context is any set of circumstances that would one set of OIC Clients from another
2290 for the purpose of resource discovery. A context is usually determined at deployment and from
2291 application requirements. An example of a context could be a multicast group- an OIC Device that
2292 is a member of more than one multicast group may have to find and select a RD in each of the

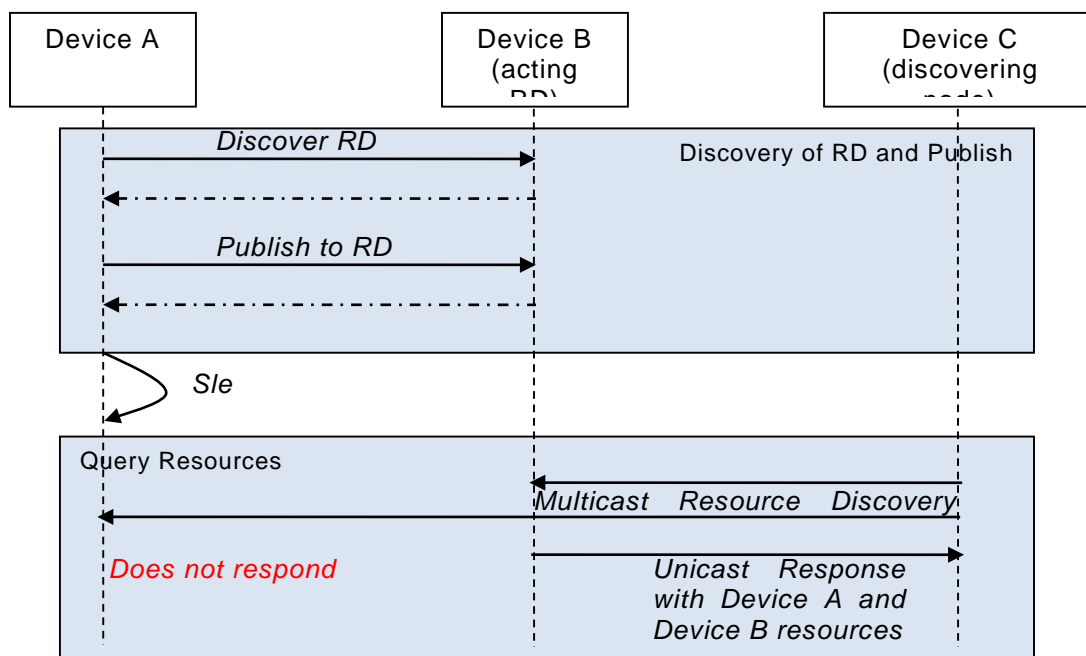
2293 multicast groups (i.e. per context) to publish its information. The OIC Device may decide to choose
 2294 other RDs during its lifetime but at no instance shall the OIC Device publish its resource information
 2295 to more than one RD per context. Devices such as TV, network router, desktop will have higher
 2296 weightage or bias factor compared to mobile phone device.

2297 This remainder of this section is divided into two parts first part covers discovering of the RD and
 2298 publishing, updating and deleting of resources for the constrained/sleepy device. Second part
 2299 covers where RD replies to queries from devices looking to discover resources.

2300 **11.3.6.2 Resource directory discovery**

2301 **11.3.6.2.1 Discovering a resource directory**

2302 A RD in the OIC network shall support RD discovery, shall provide the facility to allow devices to
 2303 publish their resource information to a RD, to update resource information in a RD and to delete
 2304 resource information from a RD.



2305

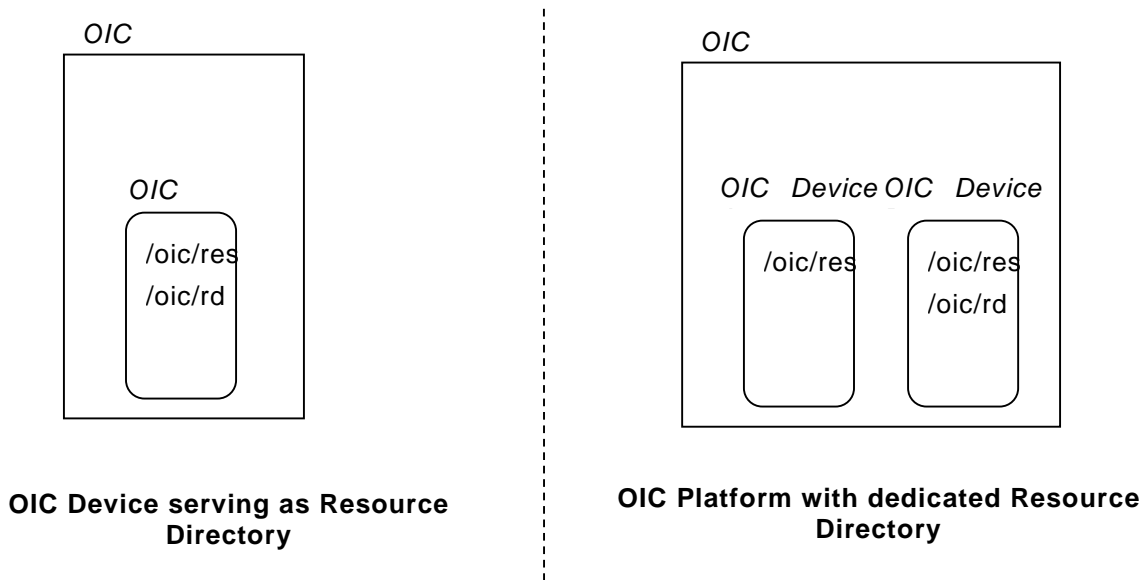
2306 **Figure 28. RD discovery and RD supported query of resources support**

2307 As shown in Figure 28, the OIC Device that wishes to advertise its resources: first discovers a
 2308 resource directory and then publishes the desired resource information. Once a set of resources
 2309 have been published to a RD then the publishing device shall not respond to multicast resource
 2310 discovery queries for those published resources when the RD is on the same multicast domain. In
 2311 that case, only the RD shall respond to multicast resource discovery requests on the resource
 2312 published to it.

2313 An OIC network allows for more than one device acting as a RD. The reason to have multiple RD
 2314 support is to make network scalable, handle network failures and centralized device failure
 2315 bottleneck. This does not preclude a scenario where a use case or deployment environment may
 2316 require single device in the environment to be deployed as the only resource directory (e.g.
 2317 gateway model). There may be more than one OIC Device acting as RD on an OIC Platform.

2318 Discovering of an RD could result in responses from more than one RD. The discovering device
 2319 shall select a RD. The selection may be based on the weightage parameter(s) provided in the
 2320 response from the RD.

2321 An RD will be application agnostic i.e., application should not be aware whether resource directory
 2322 was queried to get the resource information. All the handling of the retrieval is kept opaque to the
 2323 application. An OIC Client that performs resource discovery uses an RD just like it may use any
 2324 other OIC Server for discovery. It may send a unicast request to the RD when it needs only the
 2325 resource advertised on the RD or do a multicast query when it does not require or have explicit
 2326 knowledge of an RD.



2327
 2328 **Figure 29. Resource Direction Deployment Scenarios**

2329 Resource directory can also be discovered in the following manners:

- 2330 • Pre-configuration: Devices wishing to publish resource information may be configured a priori
 2331 with the information (e.g. IP address, port, transport etc.) of a specific resource directory. This
 2332 pre-configuration may be done at on-boarding or may be updated on the device using an out-
 2333 of-band method. This pre-configuration may be done by the manufacturer or by the user/device
 2334 manager.
- 2335 • Query-oriented: An OIC Client wanting to discover resource directories using query-oriented
 2336 discovery (i.e. pull) shall issue multicast resource discovery request directed to the /oic/rd
 2337 resource. Only and all devices that can be a RD shall host the /oic/rd and shall respond to this
 2338 query. The response shall include information about the RD (as defined by the resource type)
 2339 and weightage parameters to allow the discovering device to select between RDs (see details
 2340 in RD selection section). The /oid/rd resource is a conditionally mandatory core resource that
 2341 shall be instantiated on all OIC Devices only when offering or acting as resource directories.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://openinterconnect.org/schemas/oic.rd.selection.json#",
  "title": "RD Selection",
  "definitions": {
    "oic.rd.attributes": {
      "type": "object",
      "properties": {
        "n": {
          "type": "string",
          "description": "A human friendly name for the Resource Directory",
          "format": "UTF8"
        }
      }
    }
  },
  "di": {
```

```

    "type": "string",
    "description": "A unique identifier for the Resource Directory",
    "format": "uuid"
  },
  "sel": {
    "description": "Selection criteria that a device wanting to publish to any
RD can use to choose this Resource Directory over others that are discovered",
    "oneOf": [
      {
        "type": "object",
        "properties": {
          "pwr": {
            "type": "string",
            "enum": [ "ac", "batt", "safe" ],
            "description": "A hint about how the RD is powered. If AC then this
is stronger than battery powered. If source is reliable (safe) then appropriate
mechanism for managing power failure exists"
          },
          "conn": {
            "type": "string",
            "enum": [ "wrd", "wrls" ],
            "description": "A hint about the networking connectivity of the RD.
*wrd* if wired connected and *wrls* if wireless connected."
          },
          "cpu": {
            "type": "integer",
            "description": "Memory available at request in MHz units"
          },
          "memory": {
            "type": "integer",
            "description": "A processing capacity of the CPU specified in MB
units"
          },
          "load": {
            "type": "array",
            "items": {
              "type": "number",
            },
            "minitems": 3,
            "maxitems": 3,
            "description": "Current load capacity of the RD. Expressed as a load
factor 3-tuple (upto two decimal points each). Load factor is based on request
processed in a 10 minute window, in the last hour and hourly average over the RD
current lifetime"
          }
        }
      },
      {
        "type": "integer",
        "minimum": 0,
        "maximum": 100,
        "description": "A bias factor calculated by the Resource directory - the
value is in the range of 0 to 100 - 0 implies that RD is not to be selected. Client
chooses RD with highest bias factor or randomly between RDs that have same bias
factor"
      }
    ]
  },
  "required": [ "sel" ]
}
},

```

```
"type": "object",
"$ref": "#/definitions/oic.rd.attributes"
}
```

Figure 30. Information in a response to a query to /oic/rd

2342

2343 • Advertisement/Presence: An RD may advertise about its presence to devices. It is a
2344 combination of presence and advertisement packet. The devices that are already publishing to
2345 a RD may use this as a presence or heartbeat message of the RD. If the RD advertisement
2346 does not arrive at a stipulated interval, publishing device starts searching for other RDs in the
2347 network, as this is a signal that RD is not online. Other usage of this message is it serves as
2348 an advertisement for a device seeking a RD to publish their resources. The details from the
2349 advertisement can then be used to query directly to a RD to get weightage details instead of
2350 sending a multicast packet in a network. As it is intended this is sent at a regular interval and
2351 does not include weightage information to keep packet sizes small.

2352 • One of the important benefits of an RD is to make services discoverable in networks that don't
2353 support site wide multicast but do support site wide routing. An example of such a network is
2354 Homenet .To enable an RD function across such a network a site discovery mechanism is
2355 needed to discover the RD service (IP address & port number). Homenets that support hybrid
2356 proxy (IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00) allow site wide discovery based on
2357 dns-sd/mDNS. In order to make itself discoverable beyond the link local scope, an RD with a
2358 routable ip address shall implement the mDNS responder requirements defined in
2359 IETF RFC 6762. The RD shall respond to mDNS queries of type PTR and with a service name
2360 equal to "_rd._sub._oic._udp.local". The response shall include all routable IP addresses. An
2361 OIC Devices with a routable ip address shall discover all available RD instances by issuing a
2362 DNS-SD's PTR lookup as defined in IETF RFC 6763 with as service name service name
2363 "_rd._sub._oic._udp.local". The response shall include all routable routable addresses/port pair
2364 through which the RD service is made accessible.

2365 11.3.6.2.2 Resource directory selection process

2366 11.3.6.2.2.1 Selection criteria

2367 When a device discovers more than one RD then it shall decide to use one of these RDs based on
2368 the selection criteria described here. A device shall use or publish information to only one RD
2369 within a multicast domain at a given time. This is to minimize the burden of processing duplicate
2370 information in the resource discovery phase.

2371 There two ways to select an RD. One is based on a weighting or bias factor (RD generated) and
2372 the other is based on clients determination based on granular parameters provided by the server
2373 (client/device generated). Devices may use one or both methods to select an RD.

2374 *Bias factor:* The bias factor is a server generated positive number in the range of 0 to 100, where
2375 0 is the lowest to 100 being the highest. If two RDs have the same bias factor then the selecting
2376 device may choose either based auxiliary criteria or at random. Either way only one RD shall be
2377 selected and used at a time. No specific method is defined in this specification to determine the
2378 bias factor for an RD. The number may be a pre-configured value at the time of on-boarding or
2379 subsequent configuration of the RD or may be based on a formula determined by the
2380 implementation of the RD. (OIC will provide a standard formula for this calculation in a future
2381 version or release of specification).

2382 The bias factor shall be calculated by the RD by adding the contribution values determined for
2383 each of the parameters in Table 24 and divided by the number of parameters. An RD may advertise
2384 a bias factor larger than the calculated value when there is reason to believe that the RD is highly
2385 capable for example an installed service provider gateway or point of presence.

2386 *Parameters:* Optionally, parameters defined in Table 24 (like direct power supply, network
2387 connectivity, load conditions, CPU power, memory, etc.) may be returned in the discovery

2388 response. Discovering device may use the details to make granular selection decisions based on
 2389 client defined policies and criteria that use the RD parameters. For example, a device in an
 2390 industrial deployment may not weight power connectivity high but another in home environments
 2391 may give more weightage for power.

2392

Table 24: 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) Bandwidth: <ul style="list-style-type: none"> • High (10) • Low (5) • Lossy (3) Interfaces	<ul style="list-style-type: none"> • Network product = [sum of (type * bandwidth per network interface)]/[number of interfaces] • Normalized to 100
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/ldavg1.pdf • Factor = $100 - ((L1*3 + L5*7 + L15*10)/3)$

2393

2394 **11.3.6.2.2.2 Selection scenarios**

2395 The publishing device uses advertisement and query mechanism to find about a RD. There are
 2396 four scenarios based on how selection process will work:

- 2397 1) A single or multiple RDs are already present in the network when a new device boots up.
 2398 2) No RD is present in the network and a new device boots up.
 2399 3) Another device boots up that has the capability of working as a RD.
 2400 4) Two RD boots at the same time, like after power failure.

2401 For first scenario where RD is already present, device listens to advertisement packet or queries
 2402 for RDs on the multicast address, as described above. If a single or multiple responses are
 2403 received publishing device uses the weightage information from the query response to select the
 2404 RD to publish resources to.

2405 In the second scenario, device will listen to the advertisement. Once RD advertisement packet is
2406 received it can receive the weightage information or retrieve it and if the bias meets its criteria, it
2407 can register its resources on the RD.

2408 In the third scenario, if the publishing device is publishing to an existing RD and it discovers the
2409 new RD, the publishing device may choose to move to the RD if the relative bias factors favour
2410 the new RD. If the decision is made to select the new RD, the then publishing device shall delete
2411 its resource information from the previous RD and then publish the information to the new RD. (In
2412 the transition period, once the delete request has been sent, the publishing device shall respond
2413 to resource discovery requests).

2414 In the last scenario, each RD starts and advertises about their presence. Publishing device based
2415 on the weightage criteria selects appropriate a RD for publishing its resource information.

2416 **11.3.6.3 Resource publishing**

2417 **11.3.6.3.1 Publish resources**

2418 **11.3.6.3.1.1 Overview**

2419 After the selection process of a RD, a device may choose one of the following mechanisms:

- 2420 • Push its resources information to the selected RD or
- 2421 • Request the RD to pull the resource information by doing a unicast discovery request against
2422 its /oic/res

2423 The publishing device may decide to publish all resources or few resources on the resource
2424 directory. The publishing device shall only publish resources that are otherwise published to its
2425 own /oic/res. A publishing device may respond to discovery requests (on its /oic/res resource) for
2426 the resources it does not publish to a RD. Nonetheless, it is highly recommended that when an RD
2427 is used, all discoverable resources on the publisher be published to the RD.

2428 **11.3.6.3.1.2 Publish: Push resource information**

2429 Resource information is published using an UPDATE CRUDN operation to /oic/rd using the
2430 resource type oic.wk.rdpub and the oic.if.baseline interface.

2431 Once a publishing device has published resources to a RD, it may not respond to the multicast
2432 discovery queries for the same resources against its own /oic/res, especially when on the same
2433 multicast domain as the RD. After publishing resources, it is a RD responsibility to reply to the
2434 queries for the published resources.

2435 If the publishing device is in sleep mode and a RD has replied on behalf of the publishing device,
2436 then a discovering device will try to access resource on the provided URI.

2437 There is another possibility that the resource directory and the publishing device both respond to
2438 the multicast query from the discovering device. This will create a duplication of the packet but is
2439 an alternate that may be used for non-robust network. It is not a recommended option but for
2440 industrial scenarios, this is one of the possibilities. Either way, discovering clients shall always be
2441 prepared to process duplicate information in responses to multicast discovery request.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://openinterconnect.org/schemas/oic.rd.publish.json#",
  "title": "RD Publish & Update",
  "definitions": {
    "oic.rd.publish": {
      "type": "object",
      "description": "Publishes resources as OIC Links into the resource
directory",

```



```

        "properties": {
            "n": {
                "type": "string",
                "description": "Readonly, Human friendly name of the publishing OIC
Device"
            },
            "id": {
                "type": "string",
                "description": "ReadOnly, Unique identifier (UUID) for device that is
publishing",
                "format": "uuid"
            },
            "$ref":
"oic.collection.json#/definitions/oic.collection.setoflinks/properties",
            "lt": {
                "type": "integer",
                "description": "Time to indicate a RD, how long to keep this published
item. After this time (in seconds) elapses, the RD invalidates the links. To
keep link alive the publishing device updates the ttl using the update schema"
            }
        },
        "dependencies": {
            "links": [ "lt" ]
        }
    },
    "type": "object",
    "$ref": "#/definitions/oic.rd.publish",
    "required": [ "di" ]
}

```

2442 **Figure 31. Publish – push resource information**

2443 **11.3.6.3.2 Update resource information**

2444 Server will hold the publish resource information till the time specified in the ttl field. A device can
2445 send update if it seeks a RD to keep holding resources and reply to queries on its behalf. Update
2446 can be used for updating about all resources that are published on a RD or can use to do per
2447 resource published.

2448 Updates are done using the same resource type and interface as for the initial publish but only the
2449 information to be updated is provided in the payload.

2450 **11.3.6.3.3 Delete resource information**

2451 A resource information hold at the resource directory can be removed anytime by the publishing
2452 device. It can be either for the whole device information or for a particular resource. This resource
2453 should be only allowed when device meets a certain requirement, as it can create potential security
2454 issue.

2455 The delete is done using the device ID “id” as the tag in DELETE request query when all the
2456 resource information from the device is to be deleted. In the case of a specific resource then the
2457 DELETE request shall include the instance “ins” tag along with the device ID in the query.

2458 Selective deletion of information for individual resources is not possible the case where the RD
2459 pull the resource information. The publishing device can request a delete but only for all the
2460 resource information that the RD has pulled from that device. In this case, the DELETE request
2461 has the device ID “id” tag in the query.

2462 **11.3.6.3.4 Transfer resource information from one RD to another**

2463 When a publishing device identifies an RD that is better suited, it may decide to publish to that RD.
2464 Since the device shall publish to only one RD at a time, the client shall ensure that previously
2465 published information is deleted from the currently used RD before publishing to the newly selected
2466 RD. The deletion of the resource may be done either by allowing the TTL to expire or explicitly
2467 deleting the resource information.

2468 RDs shall not communicate resource information between themselves. It is the client's
2469 responsibility to choose the RD and to manage the published resources.

2470 **11.3.6.4 Resource discovery**

2471 **11.3.6.4.1 Query and retrieving of the resources**

2472 The query based discovery process remains the same as that in the absence of an RD. Resources
2473 may be discovered by querying the /oic/res resource by sending a multicast or unicast request. In
2474 the case of a multicast discovery request, an RD will respond for the device that hosts the
2475 resources. OIC Clients shall be prepared to process duplicate resource information from more than
2476 one RD responding with the same information or from an RD and the hosting device (publishing
2477 the resource information) both responding to the request. Interaction with resources discovered
2478 using the RD is done using the same mechanism and methods as with resources discovered by
2479 querying the /oic/res resource of the device hosting the resources (e.g., connect to the resource
2480 and perform CRUDN operations on the resource).

2481 **11.3.6.4.2 Security considerations**

2482 Resource directory should support DTLS. Communication between device and resource directory
2483 should be based on the DTLS where ever possible. The two ends, device and resource directory
2484 should be authenticated using PSK, certificates and raw public key where ever possible.

2485 The device should communicate with resource directory using the UUID after registration i.e.
2486 retrieval, update and delete should use UUID as the IP address can change where the resource
2487 are located and will confuse the resource directory.

2488 Access control list, should be used for publish and lookup purpose as they might differ. Lookup
2489 should check where the request originates from, network or resource level.

2490 Resource directory when a wild card lookup is requested should only be returned where routing is
2491 possible to check to avoid DDoS attacks. As with UDP it will not be possible to check routing, wild
2492 card lookup should be supported only over DTLS or TCP or TLS.

2493 **11.4 Notification**

2494 **11.4.1 Overview**

2495 An OIC Server shall support NOTIFY operation to enable an OIC Client to request and be notified
2496 of desired states of one or more OIC Resources in an asynchronous manner. Section 11.4.2
2497 specifies the observe mechanism in which updates are delivered to the requester.

2498 **11.4.2 Observe**

2499 In observe mechanism the OIC Client utilizes the RETRIEVE operation to require the OIC Server
2500 for updates in case of OIC Resource state changes. The Observe mechanism consists of five
2501 steps which are depicted in Figure 32 and described below.

2502 Note: the observe mechanism can only be used for a resource with a property of observable
2503 (section 7.1.4.2.4).

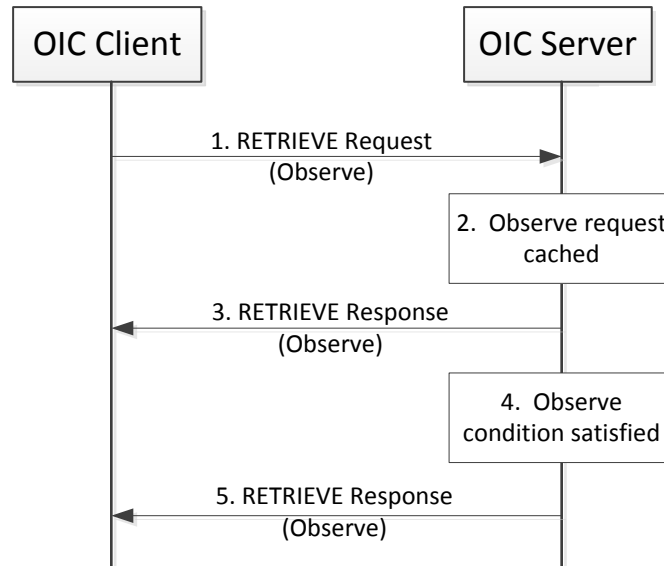


Figure 32. Observe Mechanism

11.4.2.1 RETRIEVE request with observe indication

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

- *fr*: Unique identifier of the OIC Client
- *to*: Resource that the OIC Client is requesting to observe
- *ri*: Identifier of the RETRIEVE request
- *op*: RETRIEVE
- *obs*: Indication for observe request

11.4.2.2 Processing by the OIC Server

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

11.4.2.3 RETRIEVE response with observe indication

The OIC Server shall transmit a RETRIEVE response message in response to a RETRIEVE request message from an OIC Client. The RETRIEVE response message shall include the following parameters.

- *fr*: Unique identifier of the OIC Server
- *to*: Unique identifier of the OIC Client
- *ri*: Identifier included in the RETRIEVE request
- *cn*: Information resource representation as requested by the OIC Client
- *rs*: The result of the RETRIEVE operation
- *obs*: Indication that the response is made to an observe request

2531 **11.4.2.4 Resource monitoring by the OIC Server**

2532 The OIC Server shall monitor the state the OIC Resource identified in the observe request from
 2533 the OIC Client. Anytime there is a change in the state of the observed resource, the OIC Server
 2534 sends another RETRIEVE response with the observe indication.

2535 **11.4.2.5 Additional RETRIEVE responses with observe indication**

2536 The OIC Server shall transmit updated RETRIEVE response messages following observed
 2537 changes in the state of the OIC Resources indicated by the OIC Client. The RETRIEVE response
 2538 message shall include the parameters listed in section 11.4.2.3.

2539 **11.4.2.6 Cancelling Observe**

2540 The OIC Client which does not want to receive any more responses shall not confirm the received
 2541 response in which case OIC Server assumes that the OIC Client is no more interested in the
 2542 response and cancels Observe for that OIC Client. Additionally, the OIC Client can explicitly cancel
 2543 observe by sending a RETRIEVE request without observe field to the same resource on OIC Server
 2544 which it was observing.

2545 **11.5 Device management**

2546 The OIC Device Management includes the following functions:

- 2547 • Provisioning (On-Boarding and Configuration)
- 2548 • Monitoring
- 2549 • Diagnostics and maintenance

2550 The device management functionalities specified in this version of specification are intended to
 2551 address the basic device management features. Addition of new device management features in
 2552 the future versions of the specification is expected.

2553 **11.5.1 Monitoring**

2554 Monitoring in OIC Framework is used for monitoring the current state of the OIC Devices, typically
 2555 to check their functionality state and to ensure the OIC device is operating as expected. Monitoring
 2556 includes periodic device availability check and/or device health check.

2557 If Monitoring is supported by an OIC Device, the OIC Core Resource /oic/mon shall be supported
 2558 as the designated monitoring resource as described in Table 25.

2559 **Table 25. Optional monitoring device management OIC Core Resources**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	Interfaces	Description	Related Functional Interaction
/oic/mon	Monitoring	oic.wk.mon	oic.if.r	The resource through which the OIC Device is monitored. The resource exposes properties relevant to aspects that may be monitored. The resource properties exposed by /oic/mon are listed in Table 26.	Device Management

2560

2561 Table 26 defines oic.wk.mon resource type.

2562

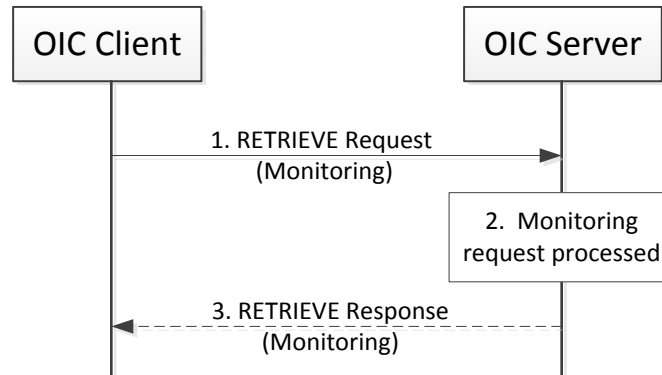
2563

Table 26. oic.wk.mon resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Availability	av	boolean	0 (not available) 1 (available)		R	yes	Indicates if the device is available or not to provide service even though it is discoverable on the network
LastedActedTime	lat	integer		sec	R	yes	Indicates the elapsed time in seconds after the device was invoked or acted upon
DeviceStatistics	ds	CSV			R	no	Contains Device Statistics Info as a CSV of integers in that order (no. of received packets, no. of sent packets)

2564

2565 An OIC Client may monitor an OIC Device by retrieving resource representation of the designated
 2566 monitoring resource hosted by an OIC Server (see Figure 33) or by retrieving information related
 2567 to a particular aspect from the designated monitoring resource (see Figure 34). Alternatively, the
 2568 OBSERVE mechanism as specified in section 11.4.2 may be used.

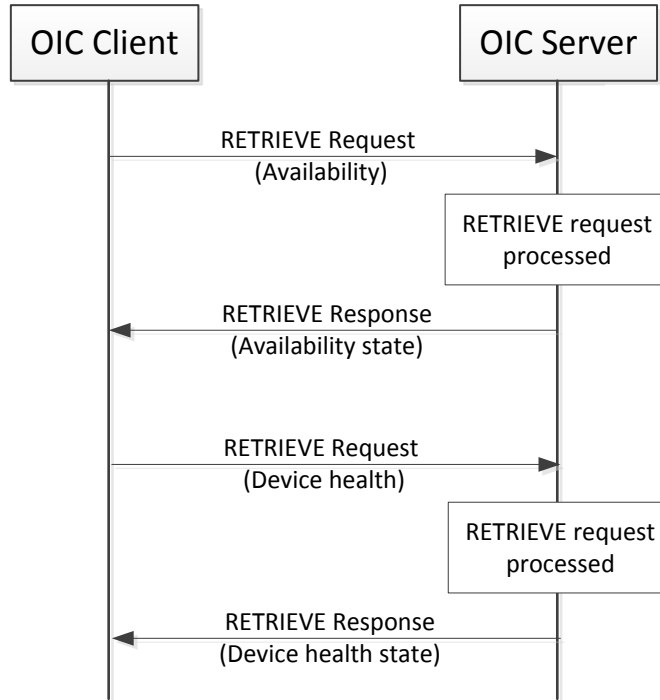


2569

2570

Figure 33. Retrieving all the monitoring information in a single request

2571



2572

2573

Figure 34. Retrieving specific Monitoring information in multiple requests

2574

11.5.2 Diagnostics and maintenance

2575

The Diagnostics and Maintenance function in OIC Framework is intended for use by the administrators to resolve issues encountered with the OIC Devices while operating in the field. If diagnostics and maintenance is supported by an OIC Device, the OIC Core Resource ‘/oic/mnt’ shall be supported as described in Table 27.

2576

2577

2578

2579

Table 27. Optional diagnostics and maintenance device management OIC Core Resources

Fixed 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 28.	Device Management

2580

2581

Table 28 defines oic.wk.mnt resource type.

2582

2583

Table 28. oic.wk.mnt resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Factory_Reset	fr	boolean			R, W	yes	0 – No action (Default*) 1 – Start Factory Reset

							After factory reset, this value shall be changed back to the default value After factory reset all configuration and state data will be lost
Reboot	rb	boolean			R, W	yes	0 – No action (Default) 1 – Start Reboot After Reboot, this value shall be changed back to the default value The reboot shall be finished within 60 seconds
StartStatCollection	ssc	boolean			R, W	Yes	0 – No collection of statistics 1 – Starts collecting statistics Toggles between collecting and not collecting any device statistics (ds property in /oic/mon) depending on the value being 0 or 1

2584

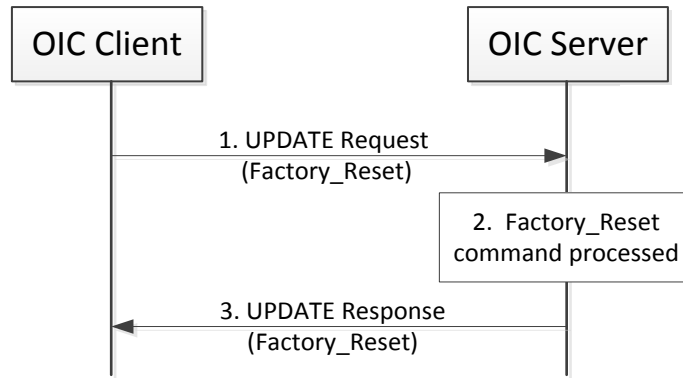
2585 Note: * - Default indicates the value of this property as soon as the device is rebooted or factory resetted

2586

2587 The OIC Framework specifies the following commands to be executed on the designated
2588 diagnostic resource of OIC Devices over the network:

- 2589 • **Factory_Reset**: Updates the device configuration to its original (default) state (factory state
2590 and equivalent to hard reboot)
- 2591 • **Start_Collection**: Triggers collection and logging information for diagnostic purposes
- 2592 • **Reboot**: Triggers a soft reboot of an OIC Device maintaining most of the configurations intact

2593 Execution of these commands may result in a change in the configuration state of an OIC Device.
2594 The configuration information in the configuration resource is expected to be updated following
2595 execution of these commands by the OIC Device, if needed. An OIC Client invokes operations on
2596 the OIC Server for executing the Diagnostic functions by sending an UPDATE message to the OIC
2597 Server. Figure 35 depicts the example of interactions between the OIC Client and the OIC Server
2598 for executing Factory Reset command.



2599
2600 **Figure 35. Factory_Reset command**

2601 **11.5.3 Security considerations for device management**

2602 Device management operations have security implications on devices. Appropriate level of security
2603 needs to be applied to all device management operations.

2604
2605 **11.6 Scenes, Rules and Scripts**

2606 **11.6.1 Introduction**

2607 Scenes, rules, and scripts are mechanisms for automating certain operations.

2608 A scene is a static entity that stores a set of defined resource property values for a collection of
2609 resources. Scenes provide a mechanism to store a setting over multiple OIC resources that may
2610 be hosted by multiple separate OIC servers. Scenes, once set up, can be used by multiple OIC
2611 clients to recall a setup.

2612 A rule is a logical “if then” statement. It consists of a rule condition and a Rule Member (a script).
2613 The rule condition is an evaluation criterion which can include evaluation of the value of a sensor
2614 on an OIC Server. When the evaluation criterion is evaluated true then the Rule Members are set
2615 to a specific determined value. A rule condition is evaluated when one of the observed resources
2616 in the rule condition changes.

2617 A script is a programmatic element that can be used to incorporate conditionals, delays, loops and
2618 other programmatic devices, including reading and writing scenes. Scripts can consist of a set of
2619 steps that are executed either upon meeting the conditions of a rule or as part of another script, in
2620 order to automate tasks. Scripts can also be used to set a scene to a specific value. For the
2621 purposes of this specification a Script is realized as the set of Rule Members that are executed
2622 when a rule condition holds true.

2623 Rules, scripts and scenes can be grouped and reused:

- 2624 • A group of scenes is also a scene.
- 2625 • A group of scripts is also a script. A script can call other scripts (similar to subroutines) and
2626 read and set scenes.
- 2627 • A group of rules is useful for setting up scenarios or modes. For example, a vacation scenario
2628 may include a random lighting rule, and day rule, a feed the dog rule, etc.

2629 In short:

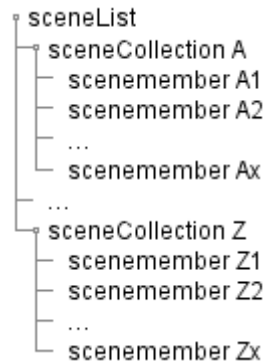
- 2630 • Scenes are bundled user settings
- 2631 • Scripts are automated background tasks
- 2632 • Rules are conditional statements that execute scripts when the condition is true

2633 **11.6.2 Scenes**

2634 **11.6.2.1 Introduction**

2635 Scenes are described in OIC by means of resources. The scene resources are hosted by an OIC
2636 Server and the top level resource is listed in /oic/res. This means that an OIC Client can determine
2637 if the scene functionality is hosted on an OIC Server via a RETRIEVE on /oic/res or via resource
2638 discovery. The setup of scenes is driven by OIC Client interactions. This includes creating new
2639 scenes, and mappings of OIC Server resource properties that are part of a scene.

2640 The scene functionality is created by multiple resources and has the structure depicted in Figure
2641 36. The sceneList and sceneCollection resources are overloaded collection resources. The
2642 sceneCollection contains a list of scenes. This list contains zero or more scenes. The
2643 sceneMember resource contains the mapping between a scene and what needs to happen
2644 according to that scene on an indicated resource.

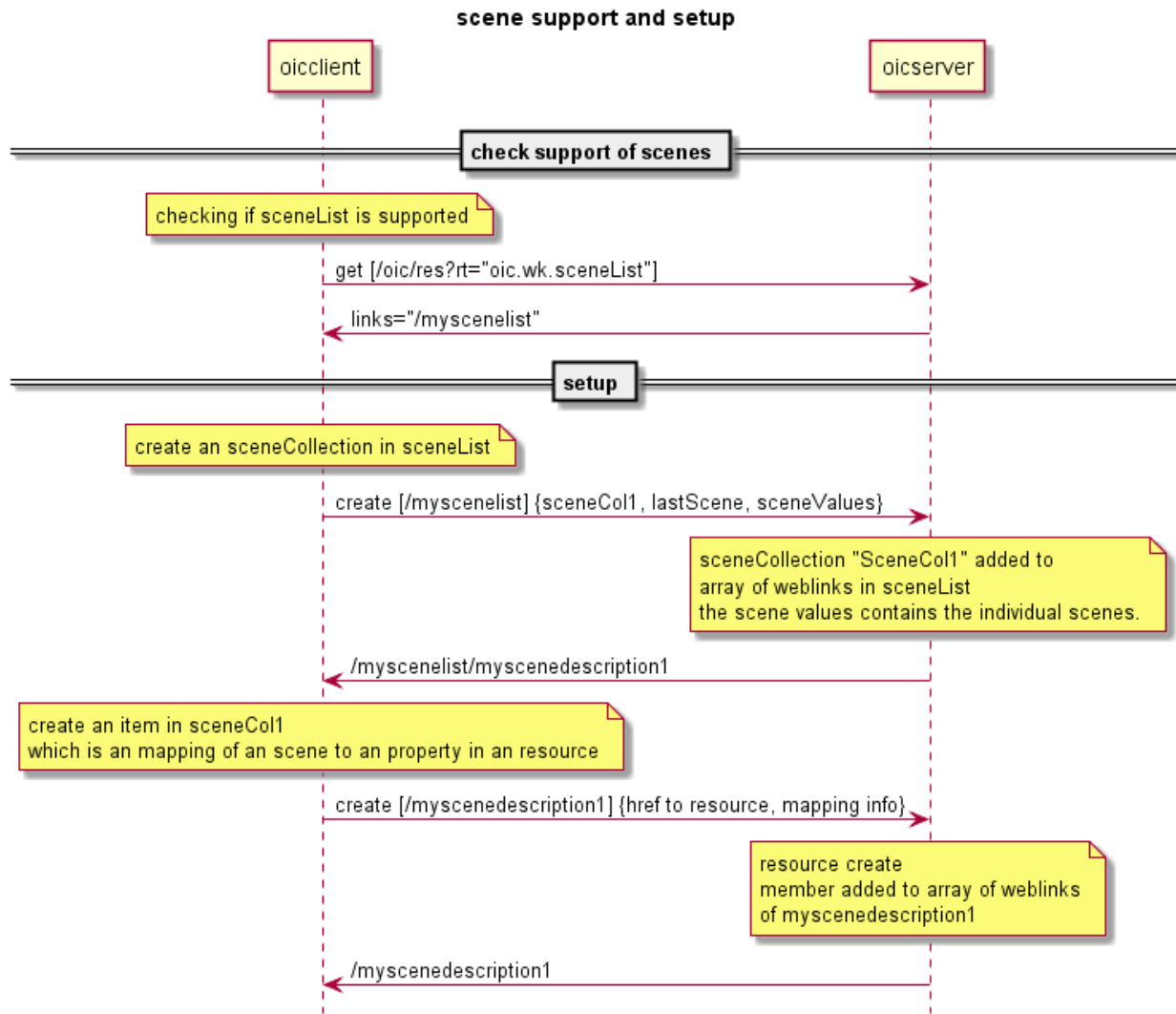


2645

2646 **Figure 36 Generic scene resource structure**

2647 **11.6.2.2 Scene creation**

2648 An OIC Client that wants to create scenes needs to verify that an OIC server supports the scene
2649 feature; the sceneMembers of a scene do not have to be co-located on the OIC server supporting
2650 the scene feature. This can be done by checking if /oic/res contains the rt of the sceneList resource.
2651 This is depicted in first steps of Figure 37. The sceneCollection can be created by an end user
2652 using a capable OIC Client. This will entail defining the scene with an applicable list of scene
2653 values and the mappings for each OIC Resource being part of the scene. The mapping for each
2654 resource being part of the sceneCollection is described by a resource called sceneMember. The
2655 sceneMember resource contains the link to a resource and the mapping between the scene listed
2656 in the sceneValues property and the actual resource property value of the OIC Resource indicated
2657 by the link.

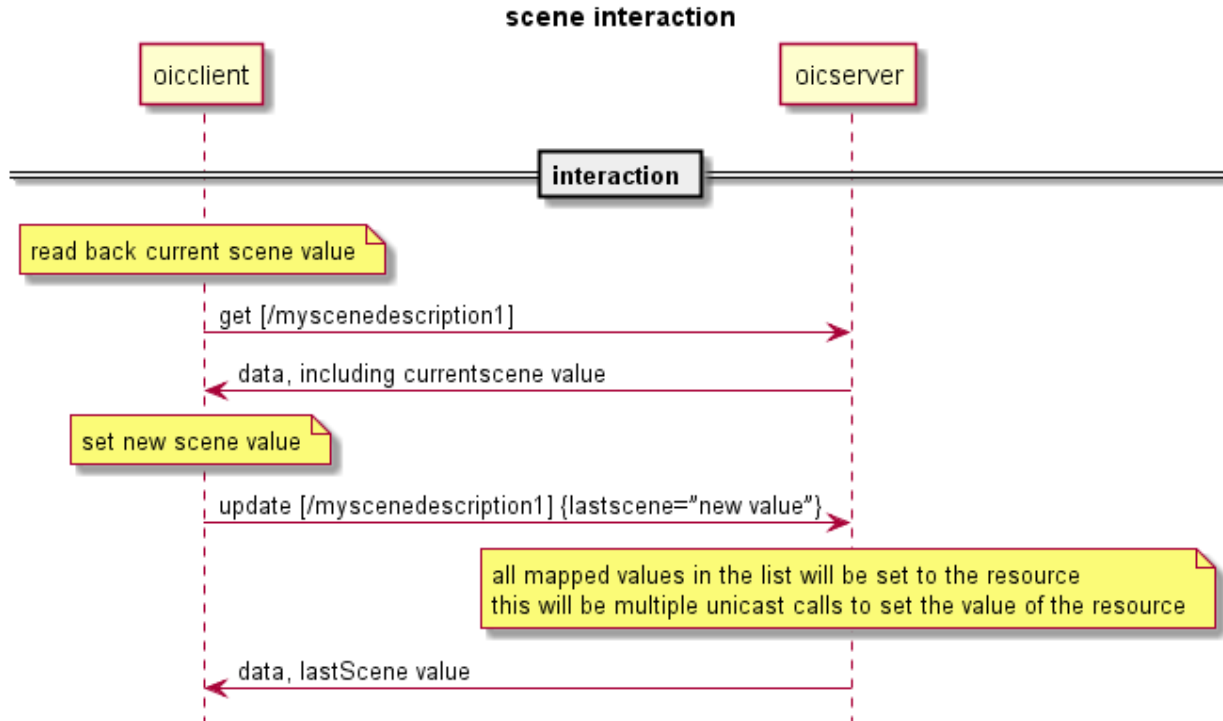


2658

2659 **Figure 37 Interactions to check Scene support and setup of specific scenes**

2660 **11.6.2.3 Interacting with Scenes**

2661 All capable OIC Clients can interact with scenes. The allowed scene values and the last applied
 2662 scene value can be retrieved from the OIC server hosting the scene. The scene value shall be
 2663 changed by issuing an UPDATE operation with a payload that sets the lastScene property to one
 2664 of the listed allowed scene values. These steps are depicted in Figure 38. Note that the lastScene
 2665 value does not imply that the current state of all resources that are part of the scene will be at the
 2666 mapped value. This is due to that the setting the scene values are not modelled as actual states
 2667 of the system. This means that another OIC Client can change just one resource being part of the
 2668 scene without having feedback that the state of the scene is changed.

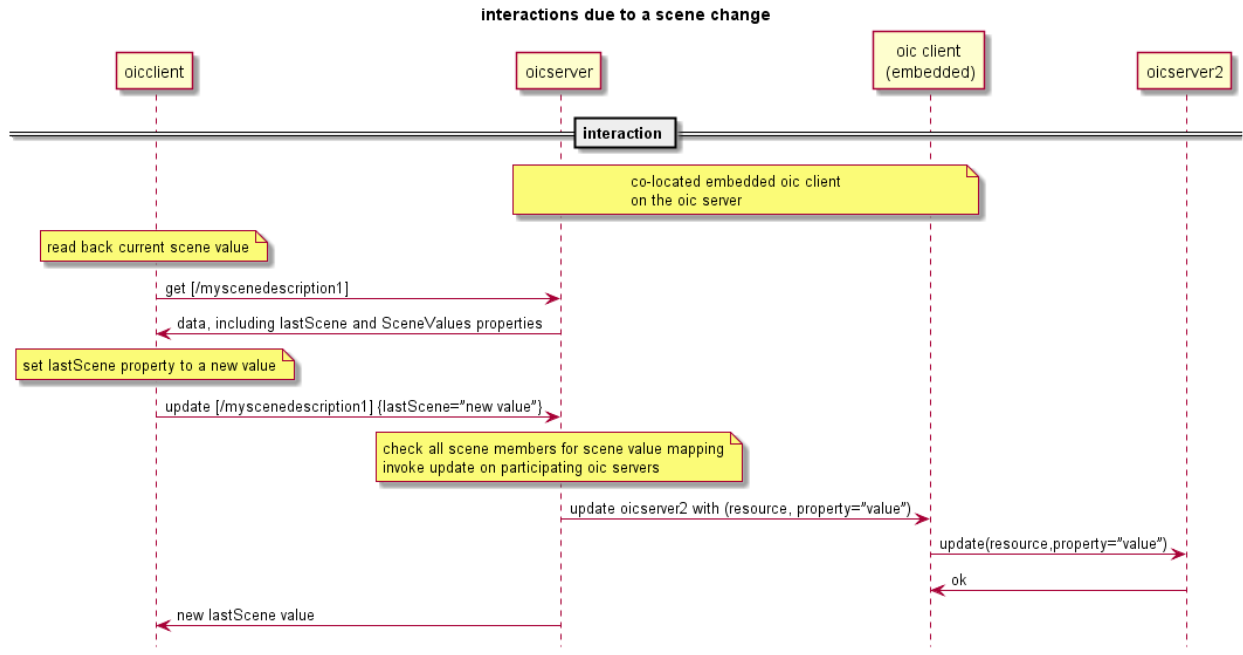


2669

2670

Figure 38 Client interactions on a specific scene

2671 As described previously, a scene can reference one or more resources that are present on one or
 2672 more OIC Servers. The scene members are re-evaluated each time a scene change takes place.
 2673 This evaluation is triggered by an OIC Client that is either embedded as part of the OIC Server
 2674 hosting the scene, or separate to the server having knowledge of the scene via a RETRIEVE
 2675 operation, observing the referenced resources using the mechanism described in section 11.4.2.
 2676 During the evaluation the mappings for the new scene value will be applied to the OIC Server. This
 2677 behaviour is depicted in Figure 39.



2678

2679

Figure 39 Interaction overview due to a Scene change

2680

11.6.2.4 Deletion of a Scene

2681

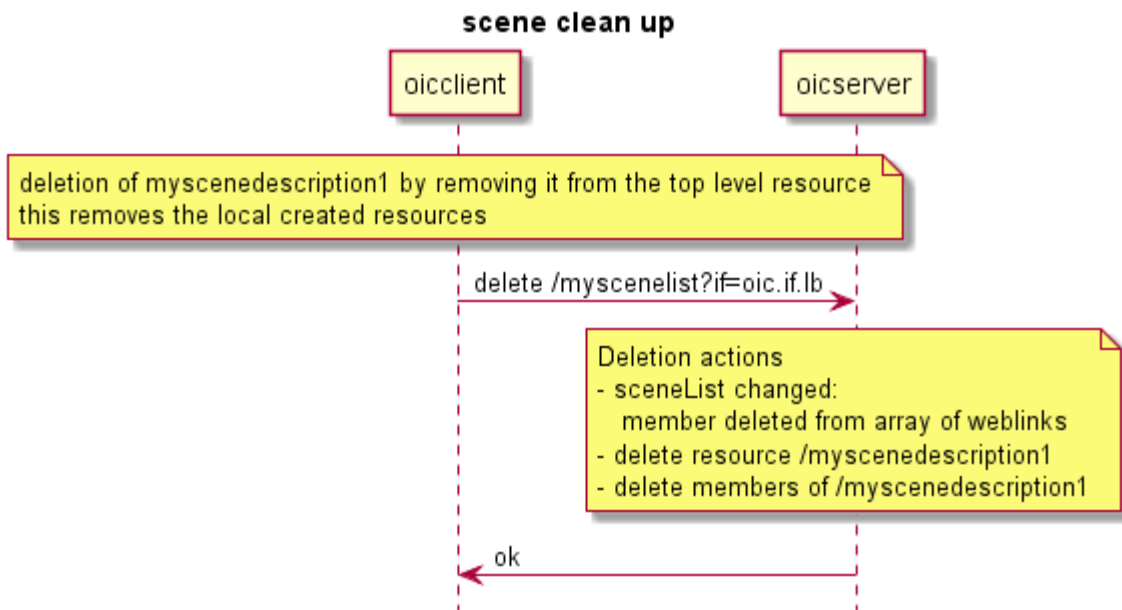
When the sceneCollection is not needed anymore the end user can remove the sceneCollection from the OIC Server. As the sceneCollection is a specialization of an OIC Collection as defined in section 7.1.6.3, then it is deleted by using the mechanism defined for deletion of collections in section 7.1.6.3.7. Note that this also deletes all contained sceneMember resources. The deletion is depicted in Figure 40.

2682

2683

2684

2685



2686

2687

Figure 40 Clean up of Scene resource structure

2688 **11.6.2.5 Summary of resource types defined for Scene functionality**

2689 Table 29 summarizes the list of resource types that are part of Scenes.

2690 **Table 29 list of resource types for Scenes**

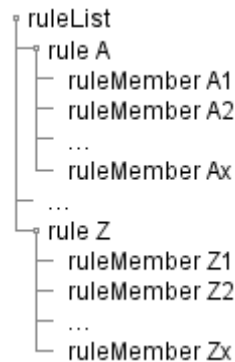
Friendly Name (informative)	Resource Type (rt)	Short Description	Section
sceneList	oic.wk.sceneList	Top Level collection containing sceneCollections	
sceneCollection	oic.wk.sceneCollection	Description of zero or more scenes	
sceneMember	oic.wk.sceneMember	Description of mappings for each specific resource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

2691 **11.6.3 Rules**

2692 **11.6.3.1 Introduction**

2693 Rules functionality (or conditional scripting) is described in OIC by means of resources. The rule
 2694 resources are hosted by an OIC Server and the top level resource is listed in /oic/res. This means
 2695 that an OIC Client can determine if the rule functionality is hosted on an OIC Server via a
 2696 RETRIEVE on /oic/res or via resource discovery. The setup of rules is done by OIC Client
 2697 interactions. This includes creating new rule resources and mappings of OIC Server resource
 2698 properties. A rule resource consist of a rule condition that evaluates and when the evaluation is
 2699 true, executing the mappings that are in the script part of the resource (in essence the set of Rule
 2700 Mmembers)

2701 The Rule functionality is created by multiple resources and has the structure depicted in Figure 41.
 2702 The ruleList and rule resources are overloaded collection resources.



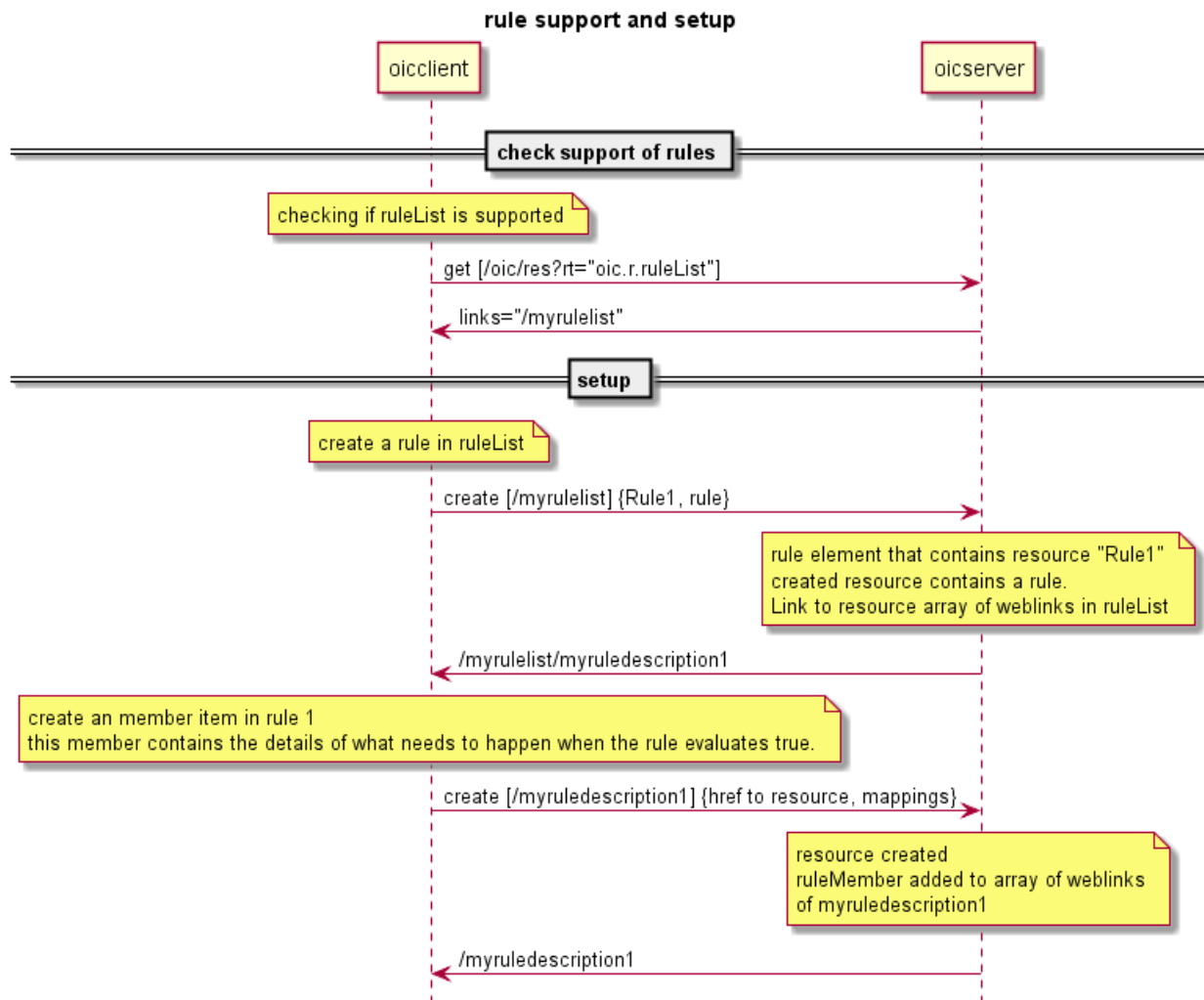
2703
2704

2705 **Figure 41 Generic rule resource structure**

2706 **11.6.3.2 Rule creation**

2707 An OIC Client that wants to create rule resources needs to verify that an OIC server supports the
 2708 rule feature. This can be done by checking if /oic/res contains the rt of the ruleList resource. This
 2709 is depicted in first steps of Figure 42. The rule resources can be created by an end user using a
 2710 capable OIC Client. This will entail creating a resource that defines a rule. The script (Rule
 2711 Members) will be executed when the rule condition evaluates to true. The rule condition is defined
 2712 by the EBNF defined in 5.6.16.1 of UPnP AV CDS. The property in the EBNF indicates a variable
 2713 which will be mapped to a property in a resource on an OIC Server. The syntax to indicate the
 2714 reference is defined as: <deviceurn>:<resourceid>:<resource properties>.

2715 An example of rule condition that references resources is:
 2716 (uuid:mybinaryid:value = true) and (uuid:myid:temperature > 30)



2717
 2718 **Figure 42 Interactions to check Rule support and setup of specific rules**

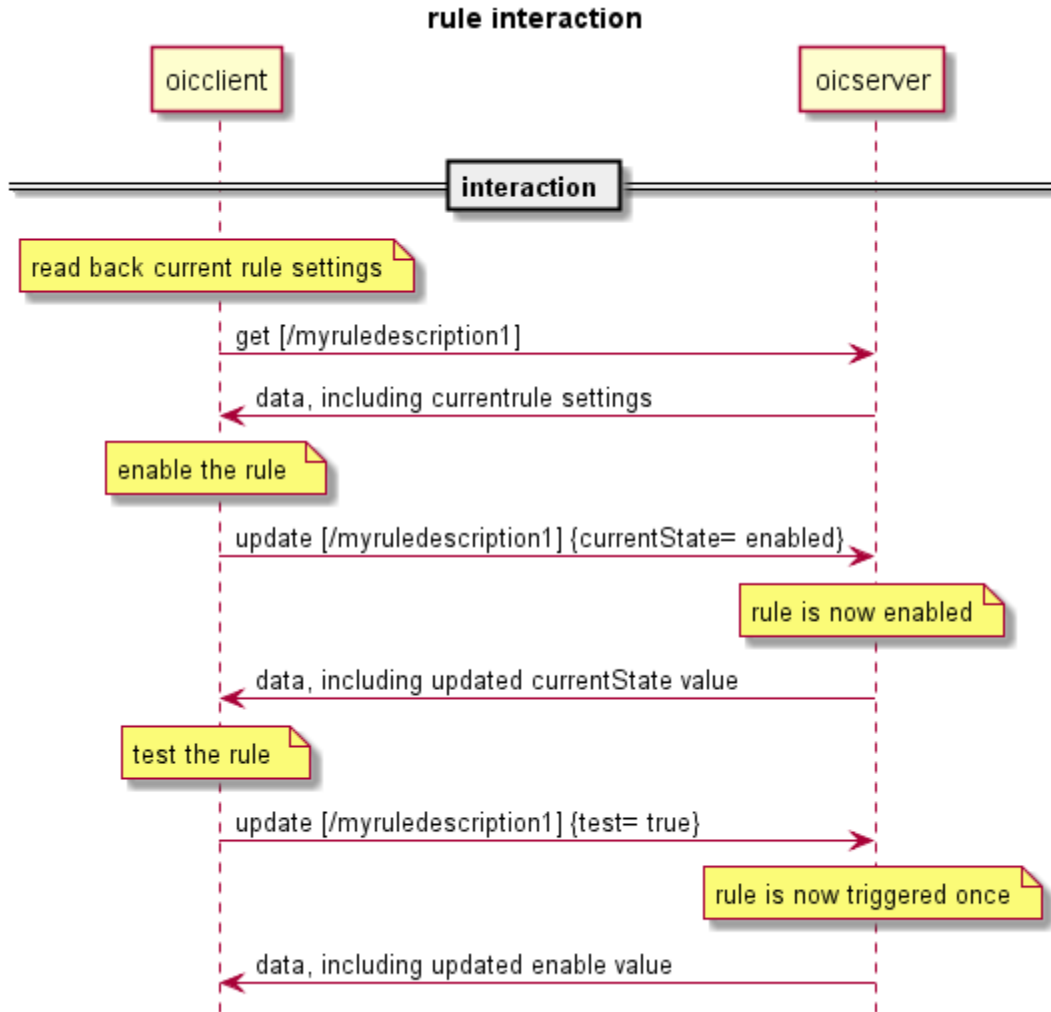
2719 **11.6.3.3 Interacting with Rules**

2720 All capable OIC Clients can interact with rules. However there are only two interactions for OIC
 2721 Clients since the rules are set up as a background process. The two interactions are:
 2722 enabling/disabling the rule and activating the test mode of a rule.

2723 An OIC Client that wishes to enable a rule shall do so by initiating an UPDATE operation with a
 2724 payload that sets the currentStatus property of the rule resource (section 8.4) to 'enabled'. An OIC
 2725 Client that wishes to disable a rule shall do so by initiating an UPDATE operation with a payload
 2726 that sets the currentStatus property of the rule resource (section 8.4) to 'disabled'.

2727 An OIC Client that wishes to run the test mode of a rule shall do so by initiating an UPDATE
 2728 operation with a payload that sets the test property of the rule resource (section 8.4) to 'true'. On
 2729 completion of the test mode the rule on the OIC Server shall reset the test property to 'false'.

2730 These steps are depicted in Figure 43.



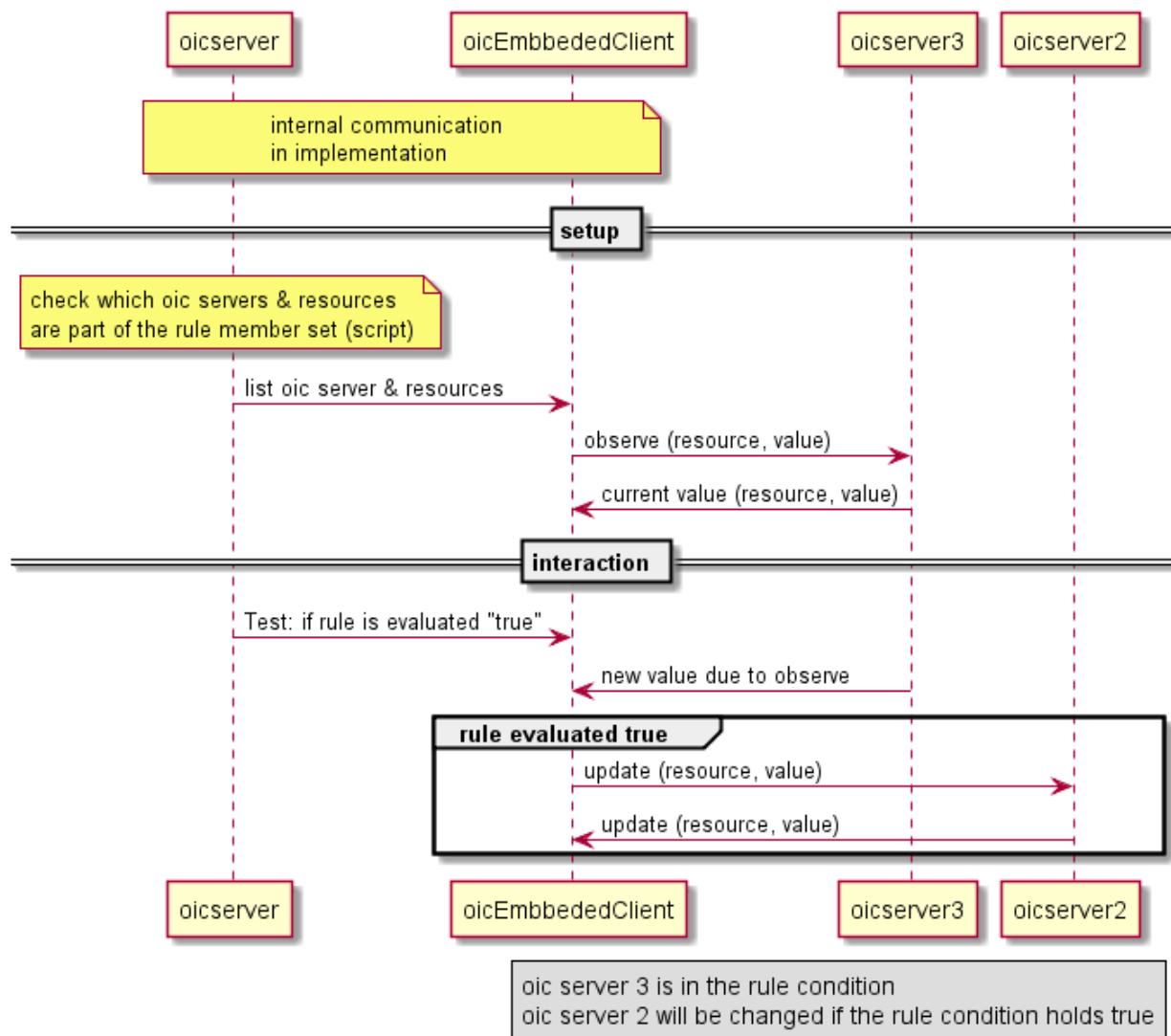
2731

2732

2733

Figure 43 Client interactions on rules

2734 As described a rule condition can reference one or more resources that are present on one or
 2735 more OIC Servers. The rule condition is re-evaluated each time the status of a resource that is
 2736 part of the condition changes. This evaluation is triggered by an OIC Client that is either embedded
 2737 as part of the OIC Server hosting the rule, or separate to the server having knowledge of the rule
 2738 via a RETRIEVE action, observing the referenced resources using the mechanism described in
 2739 Section [Ref]. If the rule condition evaluates to true then the Rule Members that are part of the
 2740 script are executed. This behaviour assuming an embedded client is depicted in Figure 44.



2741

2742

Figure 44 Interaction overview due to a rule condition evaluating to true

2743

11.6.3.4 Deletion of a Rule

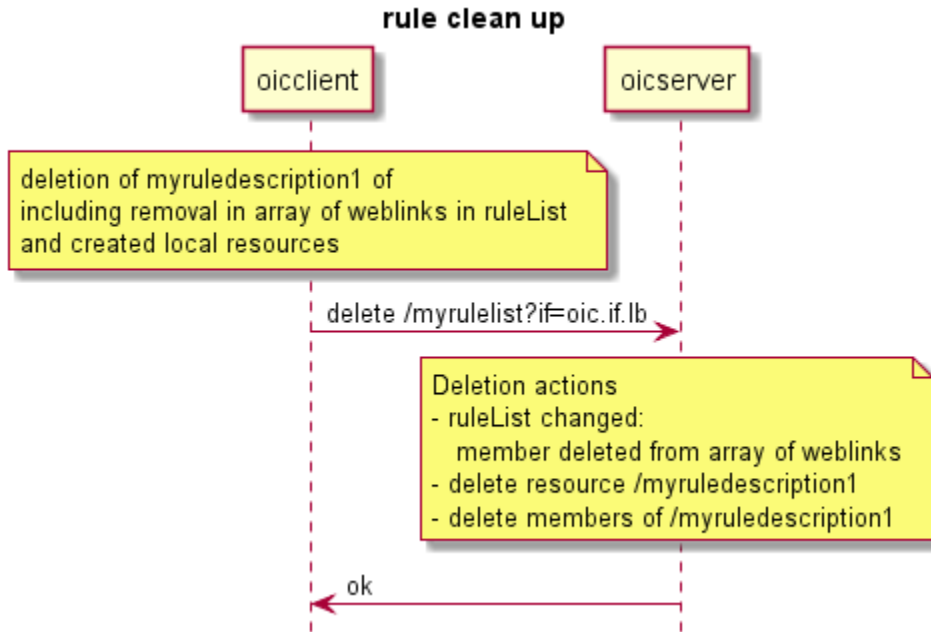
2744

2745

2746

2747

When the rule is not needed anymore the end user can remove the rule from the OIC Server. As the rule is a specialization of a Collection as defined in section 7.1.6.3 then it is deleted by using the mechanism defined for deletion of collections in section 7.1.6.3.7. Note that this also deletes all contained ruleMember resources. The deletion is depicted in Figure 45.



2748

2749

Figure 45 Clean-up of rule resource structure

2750

2751 11.6.3.5 Rules in error state

2752 The resources that are part of a rule condition or are referenced as a Rule Member may, for
 2753 whatever reason, become unavailable or inaccessible. When it is detected that a resource (either
 2754 as a result of observe or some other means) becomes unavailable the OIC Client that handles the
 2755 evaluation of the rule condition (embedded or separate) shall set the currentState of the rule on
 2756 the OIC Server to 'Error' by issuing an UPDATE operation.

2757 If on execution of test mode of a rule one or more of the UPDATE operations on the Rule Members
 2758 returns a non-success response code, then the OIC Client that handles the execution of the rule
 2759 (embedded or separate) shall set the currentState of the rule on the OIC Server to 'Error' by issuing
 2760 an UPDATE operation.

2761 11.6.3.6 Summary of resource types defined for Rule functionality

2762 Table 30 summarizes the list of resource types that are part of Rules.

2763

Table 30 List of resource types part of Rules

Friendly Name (informative)	Resource Type (rt)	Short Description	Section
ruleList	oic.wk.ruleList	Top Level collection containing rules	
rule	oic.wk.rule	Description of a rule scenario containing 1 condition and a script (set of Rule Members)	
ruleMember	oic.wk.ruleMember	Description specific resources and their values which must be set when the rule evaluates true	

2764 **11.6.4 Security considerations**

2765 Creation of Scenes and Rules on an OIC Server that is capable of this functionality is dependent
2766 on the ACLs applied to the resources and the OIC Client having the appropriate permissions.
2767 Interaction between an OIC Client (embedded or separate) and an OIC Server that hosts the
2768 resource that is referenced as a scene member or Rule Member is contingent on the OIC Client
2769 having appropriate permissions to access the resource on the host OIC Server.

2770 Reference [Ref Security] for details on the use of ACLs and also the mechanisms around Device
2771 Authentication that are necessary to ensure that the correct permissions exist for the OIC Client
2772 to access the rule or scene member resource(s) on the OIC Server.

2773

2774 **12 Messaging**

2775 **12.1 Introduction**

2776 This section specifies the protocol messaging mapping to the CRUDN messaging operations
2777 (Section 8) for each messaging protocol specified (e.g., CoAP, HTTP, etc.). Mapping to additional
2778 protocols is expected in later version of this specification. All the property information from the
2779 resource model shall be carried within the message payload. This payload shall be generated in
2780 the resource model layer and shall be encapsulated in the data connectivity layer. The message
2781 header shall only be used to describe the message payload (e.g., verb, mime-type, message
2782 payload format), in addition to the mandatory header fields defined in messaging protocol (e.g.,
2783 CoAP, HTTP) specification. If the message header does not support this, then this information
2784 shall also be carried in the message payload. Resource model information shall not be included in
2785 the message header structure unless the message header field is mandatory in the messaging
2786 protocol specification.

2787 **12.2 Mapping of CRUDN to CoAP**

2788 **12.2.1 Overview**

2789 An OIC Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in
2790 section 12.2.2. An OIC Device implementing CoAP shall conform to IETF draft-ietf-core-observe-
2791 16 to implement the CoAP Observe option. Support for CoAP block transfer when the payload is
2792 larger than the MTU is defined in section 12.2.5..

2793 **12.2.2 Request methods**

2794 Every request has a CoAP method that qualifies the request. The primary methods and their
2795 meanings are shown in Table 31, which presents the way to utilize GET/PUT/POST/DELETE
2796 methods for CREATE, RETRIEVE, UPDATE, DELETE operations. The descriptions are generic
2797 behaviours using these methods but resource interfaces may modify these generic semantics (the
2798 Default Interface reflects the generic semantics).
2799

2800

Table 31. CoAP request methods

CRUDN	Method	General semantics (see individual use for any exceptions)
RETRIEVE	GET	RETRIEVE operation is performed with the GET method as in [RFC7252]. The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request URI. GET is a safe method and is idempotent. GET is also used for introspection of a resource
CREATE UPDATE	PUT	PUT shall be used to completely replace or overwrite the entire representation of a resource. The resource representation in the payload

		<p>of the PUT request shall be the complete representation (partial representation with only a few of the properties defined is not part of PUT semantics – see POST).</p> <p>CREATE with PUT uses a new target URI for the new resource to be created. A client sends a PUT request to a server with the target URI for the new resource and the new resource representation in the payload. The server creates the new resource with the target URI provided in the PUT request and sends back a response.</p> <p>UPDATE with PUT i) uses an existing target URI for the resource to be updated and ii) carries the whole replacement as the payload. A client sends a PUT request to a server with the target URI for the resource to be updated and the representation of the total replacement in the payload. The server replaces the existing resource identified by the request URI with the representation enclosed in the PUT request and sends back a response.</p> <p>If the resource previously exists and does not have a resource representation compatible with the PUT representation in the payload (i.e. the “overwrite” semantic cannot be honoured) then an OIC Client error shall be returned.</p> <p>PUT is an unsafe method but it is idempotent which implies that when a PUT request is repeated the outcome shall be the same each time. (PUT should be a supported method on a resource or resource interaction only when idempotent behaviour can be preserved).</p>
CREATE UPDATE	POST	<p>POST may be used only in situations where the URI in the target of the request is valid – i.e. is the URI of an existing resource on the server that is processing the request. If the resource is not present, a “Resource not found error” shall be returned.</p> <p>CREATE with POST i) uses an existing target URI for the resource responsible for the creation and ii) the URI of the new resource is determined by the server, then forwarded to the client. A client sends a POST request to a server with the target URI for the resource which is responsible for creating the new resource and the representation of the new resource in the payload. Take notice that the target URI in the request is not for the new resource. The server creates the new resource with a URI, then returns the response with the new URI for the newly created resource to the requesting client.</p> <p>UPDATE with POST i) uses an existing target URI for the resource to be updated and ii) carries a partial modification as the payload. A client sends a POST request to a server with the target URI for the resource to be updated and the representation of the partial update in the payload. The server incorporates the representation enclosed in the POST request into the existing resource identified by the request URI and sends back a response.</p> <p>POST is unsafe and is the supported method when idempotent behaviour cannot be expected or guaranteed.</p>
DELETE	DELETE	<p>DELETE operation is performed with the DELETE method as in [RFC7252]. The DELETE method requests that the resource identified by the request URI be deleted.</p> <p>DELETE is unsafe but idempotent (unless URIs are recycled for new instances).</p>

2801

2802 **12.2.3 Content Type negotiation**

2803 The OIC Device framework mandates support of CBOR, however it allows for negotiation of the
 2804 payload body if more than one encoding type is supported by an implementation. In this case the

2805 accept option defined in section 5.10.4 of IETF RFC 7252 shall be used to indicate which content
2806 encodings are requested by the OIC Client.

2807 Content types supported are as shown in Table 32.

2808 **Table 32. Content Types and Content Formats**

Content Type	Content Format
application/xml	41
application/exi	47
application/json defined in IETF RFC 7159	50
application/cbor defined in IETF RFC 7049	60

2809 Note: An OIC vertical can mandate a specific content type.

2810 OIC Server and OIC Client shall send a Content-Format option every time in a message with a
2811 payload body. The Content Format option shall use the Content Format numeric value from Table
2812 32.

2813 **12.2.4 CRUDN to CoAP response codes**

2814 The mapping of CRUDN operations response codes to CoAP response codes are identical to the
2815 response codes defined in IETF RFC 7252.

2816 **12.2.5 CoAP block transfer**

2817 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT
2818 devices. However scenarios can be envisioned in which an application needs to transfer larger
2819 payloads.

2820 CoAP block-wise transfer as defined in IETF draft-ietf-core-block-187 shall be used by all OIC
2821 Servers which generate a content payload that would exceed the size of a CoAP datagram as the
2822 result of handling any defined CRUDN operation.

2823 Similarly, CoAP block-wise transfer as defined in IETF draft-ietf-core-block-187 shall be supported
2824 by all OIC Clients. The use of block-wise transfer is applied to both the reception of payloads as
2825 well as transmission of payloads that would exceed the size of a CoAP datagram.

2826 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the
2827 same reliability setting (i.e. all confirmable or all non-confirmable).

2828 An OIC Client may support both the block1 (as descriptive) and block2 (as control) options as
2829 described by IETF draft-ietf-core-block-187. An OIC Server may support both the block1 (as
2830 control) and block2 (as descriptive) options as described by IETF draft-ietf-core-block-187.

2831 **12.2.6 CoAP serialization over TCP**

2832 **12.2.6.1 Introduction**

2833 In environments where TCP is already available, CoAP can take advantage of it to provide
2834 reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For
2835 example, consider a cloud application acting as an OIC Client and the OIC Server is located at the
2836 user's home. The OIC Server which already support CoAP as a messaging protocol (e.g., Smart
2837 Home vertical profile) could easily support CoAP serialization over TCP rather than adding another
2838 messaging protocol. An OIC Device implementing CoAP Serialization over TCP shall conform to
2839 IETF draft-tschofenig-core-coap-tcp-tls-04.

2840 **12.2.6.2 Indication of support**

2841 If UDP is blocked, clients depend on the pre-configured details on the device to find support for
2842 CoAP over TCP. If UDP is not-blocked, an OIC Device which supports CoAP serialization over
2843 TCP shall populate the Messaging Protocol (mpro) property in oic/res with the value "coap+tcp" or
2844 "coaps+tcp" to indicate that the device supports messaging protocol as specified by section 11.3.4.

2845 **12.2.6.3 Message type and header**

2846 The message type transported between OIC Client and OIC Server shall be a non-confirmable
2847 message (NON). The protocol stack used in this scenario shall be as described in section 3 in
2848 IETF draft-tschofenig-core-coap-tcp-tls-04.

2849 The CoAP header as described in figure 5 in IETF draft-tschofenig-core-coap-tcp-tls-04 shall be
2850 used for messages transmitted between an OIC Client and an OIC Server. An OIC Device shall
2851 use "Alternative L3" as defined in IETF draft-tschofenig-core-coap-tcp-tls-04.

2852 **12.2.6.4 URI scheme**

2853 The URI scheme used shall be as defined in section 6 in IETF draft-tschofenig-core-coap-tcp-tls-
2854 04].

2855 For the "coaps+tcp" URI scheme the "TLS Application Layer Protocol Negotiation Extension"
2856 IETF RFC 7301 shall be used.

2857 **12.2.6.5 KeepAlive**

2858 **12.2.6.5.1 Overview**

2859 In order to ensure that the connection between an OIC Device is maintained, when using CoAP
2860 serialization over TCP, an OIC Device that initiated the connection should send application layer
2861 KeepAlive messages. The reasons to support application layer KeepAlive are as follows:

- 2862 • TCP KeepAlive only guarantees that a connection is alive at the network layer, but not at the
2863 application layer
- 2864 • Interval of TCP KeepAlive is configurable only using kernel parameters, and is OS dependent
2865 (e.g., 2 hours by default in Linux)

2866 **12.2.6.5.2 KeepAlive Mechanism**

2867 OIC Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. An OIC
2868 Server shall support a resource of type oic.wk.ping as defined in Table 33.

Table 33. Ping resource

Fixed 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 an OIC Client keeps its Connection with an OIC Server active. The resource properties exposed by /oic/ping are listed in Table 34.	KeepAlive

2870

2871 Table 34 defines oic.wk.ping resource type.

2872

Table 34. oic.wk.ping resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
Interval	in	integer	minutes		R,W	yes	The time interval for which connection shall be kept alive and not closed.

2873 The following steps detail the KeepAlive mechanisms for an OIC Client and OIC Server:

- 2874 1) An OIC Client which wants to keep the connection with an OIC Server alive shall send a PUT
2875 request to /oic/ping resource on the OIC Server updating its connection Interval.
- 2876 a) This time interval shall start from 2 minutes and increases in multiples of 2 up to a maximum
2877 of 64 minutes. It stays at 64 minutes from that point.
- 2878 2) An OIC Server receiving this ping request shall respond within 1 minute.
- 2879 3) If an OIC Client does not receive the response within 1 minute, it shall terminate the connection.
- 2880 4) If an OIC Server does not receive a PUT request to ping resource within the specified "interval"
2881 time, the OIC Server shall terminate the connection.

2882 An example of the KeepAlive mechanism is as follows:

- 2883 • OIC Client → OIC Server: PUT /oic/ping {interval: 2}
- 2884 • OIC Server → OIC Client: 2.03 valid

2885 **12.3 Mapping of CRUDN to HTTP**2886 **12.3.1 Supported HTTP features**

2887 This section defines the HTTP transport between an OIC Client and an OIC Server. The
2888 implementation of HTTP on an OIC Client and OIC Device shall conform to IETF RFC 2616]. The
2889 optional HTTP/1.1 features required by OIC are listed in this section.

2890 **12.3.2 Supported HTTP methods**

2891 HTTP/1.1 optional methods PUT, POST and DELETE methods should be supported by an OIC
2892 Client and OIC Server.

- 2893 • The OIC Server shall respond with a 405 (method Not Allowed) status code when other than
2894 HEAD, GET, PUT, POST and DELETE methods are used.

2895

2896 **12.3.3 Supported HTTP header fields**

2897 HTTP header fields (section IETF RFC 2616) are components of the message header of request
2898 and response messages. These HTTP headers define the operating parameters of an HTTP

2899 transaction. All mandated HTTP headers in IETF RFC 2616 shall be supported by the OIC Client
 2900 and OIC Server. Table 35 contains the OIC supported list of HTTP headers to request a response
 2901 for particular body payload contents. Table 35 contains the OIC supported list of HTTP headers to
 2902 signal the body content in a request and response method.

2903 The discouraged in Table 35 means that however the header is allowed it is recommended not to
 2904 use this field to conserve code space and or bandwidth.

2905 **Table 35. HTTP header fields usage in OIC**

Header Name	Used in message type	RFC Allowed Mandatory	OIC usage	Allowed values
Accept	Request	Allowed	Mandatory (if more than one allowed value is specified)	application/cbor application/json text/xml
Accept-Charset	Request	Allowed	Mandatory	UTF-8
Accept-Encoding	Request	Allowed	Allowed	chunked
Accept-Language	Request	Allowed	Discouraged	
Accept-Ranges	Request Response	Allowed	Discouraged	
Age	Response	Allowed (required for cache)	Discouraged	
Allow	Response	Mandatory (in an error response)	Mandatory(in an error response)	
Authorization	Request Response	Allowed	Allowed	
Cache-Control	Request Response	Allowed	Discouraged	
Connection	Request	Conditional Mandatory	Conditional Mandatory	
Content-Encoding	Response	Conditional Mandatory	Mandatory	

Content-Language	Response	Allowed	Discouraged	
Content-Length	Request Response	Conditional Mandatory	Conditional Mandatory	
Content-Location	Response	Allowed	Discouraged	
Content-MD5	Response	Allowed	Discouraged	
Content-Range	Response	Allowed	Allowed	
Content-Type	Request Response	Mandatory	Mandatory	application/cbor application/json text/xml
Date	Request Response	Mandatory	Mandatory	
Etag	Response	Allowed	Allowed	
Expect	Request	Allowed	Discouraged	
Expires	Response	Allowed	Discouraged	
From	Request	Allowed	Discouraged	
Host	Request	Mandatory	Mandatory	
If-Match	Request	Allowed	Discouraged	
If-Modified-Since	Request	Allowed	Discouraged	
If-None-Match	Request	Allowed	Discouraged	
If-Range	Request	Allowed	Discouraged	
If-Unmodified-Since	Request	Allowed	Discouraged	
Last-Modified	Response	Allowed	Discouraged	
Location	Response	Allowed	Allowed	

Max-Forwards	Request	Allowed	Discouraged	
Pragma	Request Response	Allowed	Discouraged	
Proxy-Authenticate	Response	Allowed		
Proxy-Authorization	Request	Allowed		
Range	Request	Allowed	Allowed	
Referrer	Request	Allowed	Discouraged	
Retry-After	Response	Allowed	Allowed	
Server	Response	Allowed	Discouraged	
TE	Request	Allowed but discouraged	Discouraged	
Trailer	Response	Allowed but discouraged	Discouraged	
Transfer-Encoding	Response	Allowed	Allowed	
Upgrade	Request	Allowed	Allowed	
User-Agent	Request	Allowed	Discouraged	
Vary	Response	Allowed but discouraged	Discouraged	
Via	Request Response	Allowed	Discouraged	
Warning	Request Response	Allowed but discouraged	Discouraged	
WWW-Authenticate	Response	Allowed	Discouraged	

2906 **12.3.4 Content Type negotiation**

2907 The OIC Device framework mandates support of CBOR, however it allows for negotiation of the
2908 payload body if more than one encoding type is supported by an implementation. In this case the
2909 accept-encoding header shall be used to indicate which content encodings are requested by the
2910 OIC Client.

2911 Note that an OIC vertical can mandate a specific content type.

2912 12.3.5 HTTP response codes

2913 OIC Devices with HTTP capabilities shall support a subset of HTTP response codes as defined in
2914 HTTP/1.1 specification IETF RFC 2616 and listed in Table 36.

2915 **Table 36. HTTP response codes**

HTTP Response Code	Description
200 (“OK”)	This response code is sent to indicate a successful transaction. This response code is often used in response to a successful GET request, with the entity-body containing a resource representation of the requested resource or object instance. Use of this response code in response to PUT, POST, or DELETE requests is discouraged, to avoid the potentially unnecessary traffic generated by returning the resource representation in the entity-body.
201 (“Created”)	This response code is sent to indicate a new object or resource has been created, at the client’s request with a PUT or POST. The Location header is used in conjunction with this response code to indicate the URI of the newly created resource. The inclusion of a resource representation of the newly created resource in the entity-body of the response is discouraged, to conserve bandwidth.
204 (“No Content”)	This response code is sent to indicate a successful transaction, but one where the response does not include an entity-body. This response code is often used in response to a successful POST request, where the resource is modified, not created. This response code is also sent in response to a successful DELETE request. This response code is also sent in response to a successful GET request, where the resource exists but has an empty representation.
400 (“Bad Request”)	This response code is used to indicate a client-side error and is used when no other 4xx response code is appropriate. Often, this response code indicates that the resource representation sent by a client with a PUT or POST is not appropriate or is malformed.
4xx	All other HTTP/1.1 defined responses in the 400 range.
500 (“Internal Server Error”)	This response code is used to indicate that the server has an internal problem and is a generic response.
5xx	All other HTTP/1.1 defined responses in the 500 range.

2916

2917 12.3.6 Method mapping

2918 The resource manipulation methods defined in this specification (CRUDN) can be matched with
2919 HTTP’s GET, POST, PUT and DELETE methods. The HTTP request methods shall be as defined
2920 in 12.2.2.

2921

2922 13 Security

2923 The details for handling security and privacy are specified in [OIC Security].

2924

2925 **14 Multi resource model support**

2926 **14.1 Interoperability issue**

2927 **14.1.1 Multiple IoT Standards**

2928 Note: Alignment and interoperability between models will be added in a later version of the
2929 specification.

2930 IoT requires standardization for interoperability among diverse devices and multiple standards are
2931 under development currently. IETF defines network and web transfer protocol (e.g. 6lowpan
2932 [RFC6775] and CoAP [RFC6690], [RFC7252]), oneM2M [oneM2M] produces technical
2933 specifications for a common M2M Service Layer [oneM2M-TS0001], [oneM2M-TS0004] and IPSO
2934 Alliance [IPSO] publishes Smart Object Guideline [IPSOSmartObjects].

2935 Multitude of IoT standards are based on "Representational State Transfer (REST)", which is a
2936 software architecture style with a coordinated set of constraints for the design of components in a
2937 distributed hypermedia system [REST]. In REST based IoT, a real world entity is represented as
2938 resource in a server, which a client accesses and manipulates the resource through
2939 representations to interact with the entity, i.e. sensing and controlling the physical environments.
2940 Moreover several IoT standards adopt the common network and web transfer protocols. oneM2M,
2941 IPSO and OIC all use CoAP and IP/ UDP, [oneM2M-TS0008], [IPSO], [OIC] so any client and
2942 server supporting those standards can exchange request and response messages.

2943 However in order to interact properly, it's not sufficient for IoT devices to be able to transfer CoAP
2944 messages. IoT devices should understand each other's resources and be aware of their semantic
2945 meaning and syntactic form. Currently each standard defines its own "resource model" and
2946 specifies a different scheme to construct resources from physical entities such as light [OIC],
2947 [IPSOFramework], [IPSOSmartObjects], [oneM2M-TS0001]. Hence client and server adopting
2948 different standards can't perform meaningful interaction, i.e. the client can't manipulate the
2949 resource representation in the server.

2950 For wider interoperability among multiple standards, IoT devices need to understand each other's
2951 resource model to process CoAP request and response message properly. To interpret resources
2952 correctly, client and server need to determine which resource model each other follows in the first
2953 place. The client should be aware of whether its corresponding server adopts oneM2M or OIC
2954 model and vice versa.

2955 **14.1.2 Different resource models**

2956 OIC specification follows a resource oriented architecture with RESTful architectural style. Without
2957 common understanding on resource model, two IoT devices can't interact with each other.

2958 Currently multiple organizations such as OIC, IPSO Alliance or oneM2M, define their own resource
2959 model in difference ways, which may restrict interoperability to the respective ecosystems. The
2960 main discrepancies are as follows

- 2961 • **Resource structure:** Some define resource to have attributes (e.g. oneM2M), whereas
2962 others define it atomic and not decomposed into attributes(e.g. IPSO alliance). For example,
2963 a smart light may be represented as a resource with on-off attribute or a resourcecollection
2964 with on-off resource. In the former, on-off attribute doesn't have URI and should be
2965 accessed indirectly via the resource. In the latter, being a resource itself, on-off resource
2966 is assigned its own URI and can be directly manipulated.
- 2967 • **Resource name & type:** Some allow resource to be named freely and indicate its
2968 characteristic with separate resource type attribute (e.g. oneM2M). Whereas others fix the
2969 name ofresource a priori and indicate its characteristic with the name itself (e.g

2970 IPSOalliance). For example, smart light can be named anyway such as 'LivingRoomLight_1'
2971 in oneM2M but should have the fixed Object name with numerical Object ID of "IPSO Light
2972 Control (3311)" in IPSO alliance. Furthermore, in consequence, it's likely that data path in
2973 URI is freely defined in the former and predetermined for the latter.

2974 • **Resource hierarchy:**Some allow resource to be organized in hierarchy so that resource
2975 includes another resource in itself with parent-child relationship (e.g. oneM2M). Whereas
2976 others mandate resource to be of flat structure and associate with other resources only by
2977 referencing their links.

2978 In addition to the above, different organizations use different syntax and have different features
2979 (e.g. resource interface), which will inhibit IoT interoperability. When IoT client and server don't
2980 understand the resource model each supports, they can't perform RESTful transaction.

2981 For example, a smart light can be represented as an IPSO Smart Object in JSON as below:

2982

```
{
  "3311": {
    "description": "IPSO light control",
    "instances": {
      "0": {
        "resources": {
          "5850": {
            "description": "On/Off",
            "value": 0
          },
          "5851": {
            "description": "Dimmer",
            "value": 70
          }
        }
      }
    }
  }
}
```

2983

2984

2985 In the above, "3311" is an "Object ID" defining object type, 0" an "Object Instance", designating
2986 one or more instances, "5850", "5851", "Resource ID", defining resource type. Also IPSO embeds
2987 resource information in data path, so "On/Off" resource has predetermined data path of
2988 "3311/0/5850" and "Dimmer" resource datapath of "3311/0/5851"

2989

2990 Whereas the same smart light may be represented in OIC as two resources.

2991

```
{
  "n": "myLightSwtich",
  "rt": "oic.r.switch.binary",
  "value": True
}
```

```
{
  "n": "myLightBrightness",
  "rt":
  "oic.r.light.brightness",
  "brightness": 70
}
```

2992

2993

2994 **14.2 A scheme to exchange resource model information**

2995 **14.2.1 A scheme to exchange resource model information**

2996 IoT devices, i.e. client and server, need to understand the resource model which their
2997 corresponding device supports to be able to interoperate each other.

2998 For the initial step, it would help for IoT devices to indicate resource model each device supports.
2999 Then client and server may choose a common resource model for interaction, or in the absence of
3000 such a common model, rely on translation between the models, possibly with the assistance of 3rd
3001 party such as intermediary. Alignment and interoperability between models will be added in a later
3002 version of the specification.

3003 This document presents a scheme for CoAP endpoints, client and server, to exchange resource
3004 model they support.

3005 First, the Internet media type and Content-Format identifier are used to indicate a specific resource
3006 model. The Internet media types can be defined to indicate the resource models, potentially with
3007 content-coding, such as "application/ips+json", then assigned numeric Content-Format identifiers
3008 such as "123123" to minimize payload overhead for CoAP usage.

3009 Second, CoAP Accept and Content-Format Option are used to exchange the Content-Format
3010 identifiers indicating the resource models which CoAP endpoints prefer or support. A client
3011 includes the CoAP Accept option to inform a server which resource model, potentially with content-
3012 encoding, is acceptable and the server returns the payload in the preferred resource model if
3013 available. The Content-Format Option indicates the resource model which the payload follows.

3014 **14.2.2 New Content-Formats (Internet Media Type) for OIC resource model**

3015 OIC is currently developing resource model for IoT services (e.g. smart home) and strives to align
3016 with other standard bodies such as oneM2M or IPSO, so that OIC devices can interoperate with
3017 the devices following different standards. For this purpose, OIC prepares a scheme to exchange
3018 resource model information as describes in the previous sections.

3019 The resource mode indication scheme requires new Internet media types and CoAP Content-
3020 Format identifiers which can indicate OIC resource models with content encoding. Hence OIC
3021 requests from IANA the new Internet media types as below.

- 3022 • **application/oic**: indicates the payload follows OIC resource model and used in CoAP
3023 Accept Option or Content-Format Option.
- 3024 • **application/oic+json**: indicates the payload follows OIC resource model in json encoding
3025 and used in CoAP Accept Option or Content-Format Option.
- 3026 • **application/oic+cbor**: indicates the payload follows OIC resource model in cbor coding
3027 and used in CoAP Accept Option or Content-Format Option.

3028

Annex A
(informative)

Operation Examples

A.1 Introduction

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

rt=oc.example.light with resource type definition as illustration in Table 37.

Table 37. oc.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

rt=oc.example.garagedoor with resource type definition as illustration in Table 38.

Table 38. oc.example.garagedoor resource type definition

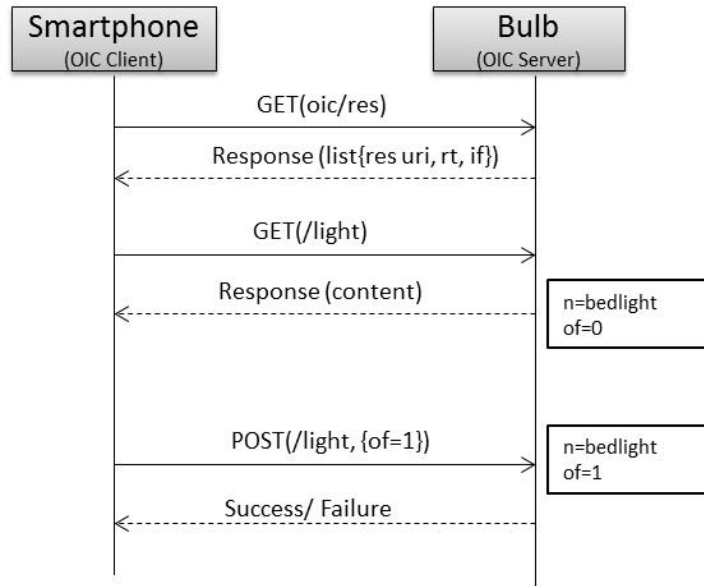
Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
open-close	oc	boolean			R, W	yes	Open/Close Control: 0 = Open 1 = Close

/oic/mon (rt=oc.wk.mon) and

/oic/mnt (rt=oc.wk.mnt) used in below examples are defined in section 7 (Resource model).

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

This sequence highlights (Figure 46) the discovery and control of an OIC light resource from an OIC smartphone.



3049

3050

Figure 46. When at home: from smartphone turn on a single light

3051 Discovery request can be sent to CoAP Multicast address 224.0.1.187 or can be sent directly to
 3052 the IP address of device hosting the light resource.

- 3053 1) Smartphone sends a GET request to '/oic/res' resource to discover all resources hosted on
 3054 targeted end point
- 3055 2) The end point (bulb) responds with the list of resource uri, resource type and interfaces
 3056 supported on the end point (one of the resource is '/light' whose rt=oic.example.light)
- 3057 3) Smartphone sends a GET request to '/light' resource to know its current state
- 3058 4) The end point responds with representation of light resource ({n=bedlight;of=0})
- 3059 5) Smartphone changes the 'of' property of the light resource by sending a POST request to '/light'
 3060 resource ({of=1})
- 3061 6) On Successful execution of the request, the end point responds with the changed resource
 3062 representation. Else, error code is returned. Details of the error codes are defined in section
 3063 12.2.4.

3064 **A.3 GroupAction execution**

3065 This example will be added when groups feature is added in later version of specification

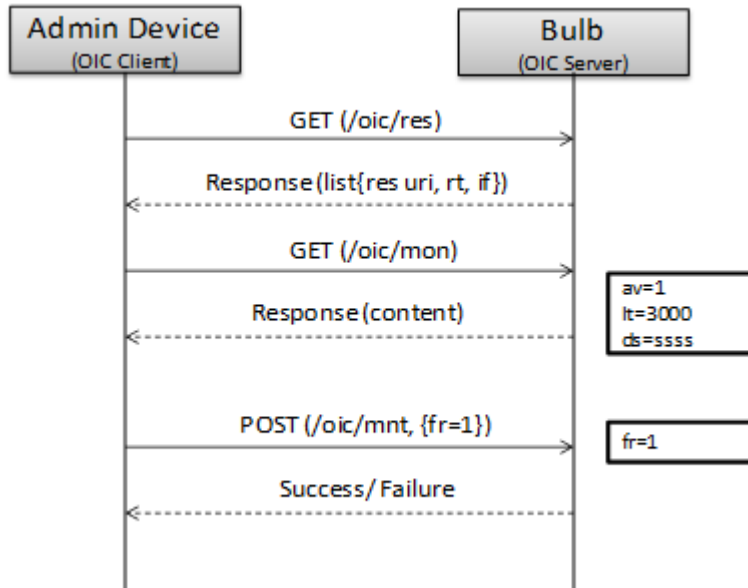
3066 **A.4 When garage door opens, turn on lights in hall; also notify smartphone**

3067 This example will be added when scripts feature is added in later version of specification

3068 **A.5 Device management**

3069 This sequence highlights (Figure 47) the device management functions of monitoring and
 3070 maintenance.

3071



3072

3073

Figure 47. Device management (monitoring and maintenance)

3074 **Pre-Condition:** Admin device has different security permissions and hence can perform device
3075 management operations on the OIC Device

- 3076 1) Admin device sends a GET request to '/oic/res' resource to discover all resources hosted on a
3077 targeted end point (in this case Bulb)
- 3078 2) The end point (bulb) responds with the list of resource uri, resource type and interfaces
3079 supported on the end point (one of the resource is '/oic/mon' whose rt=oc.wk.mon and another
3080 resource is '/oic/mnt' whose rt=oc.wk.mnt)
- 3081 3) Admin device sends a GET request to '/oic/mon' resource to know its current monitoring state
- 3082 4) The end point responds with representation of monitoring resource ({av=1;lt=3000;ds=sss...})
- 3083 5) After seeing the device statistics, Admin Device changes the 'fr' property of the maintenance
3084 resource by sending a POST request to '/oic/mnt' resource ({fr=1}). This triggers a factory reset
3085 of the end point (bulb)
- 3086 6) On successful execution of the request, the end point responds with the changed resource
3087 representation. Else, error code is returned. Details of the error codes are defined in section
3088 12.2.4.

3089
3090
3091
3092

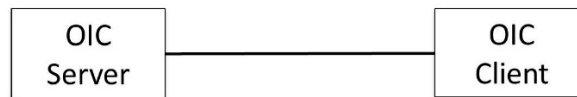
Annex B (informative)

OIC interaction scenarios and deployment models

B.1 OIC interaction scenarios

3093
3094 An OIC Client connects to one or multiple OIC Servers in order to access the resources provided
3095 by those OIC Servers. The following are scenarios representing possible interactions among OIC
3096 Roles:

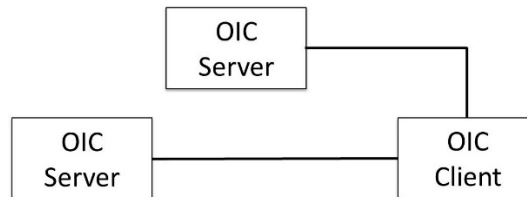
- 3097 • Direct interaction between OIC Client and OIC Server (Figure 48). In this scenario the OIC
3098 Client and the OIC Server directly communicate without involvement of any other OIC Device.
3099 A smartphone which controls an actuator directly uses this scenario.



3100

3101 **Figure 48. Direct interaction between OIC Server and OIC Client**

- 3102 • Interaction between OIC Client and OIC Server using another OIC server (Figure 49). In this
3103 scenario, another OIC Server provides the support needed for the OIC Client to directly access
3104 the desired resource on a specific OIC Server. This scenario is used for example, when a
3105 smartphone first accesses a discovery server to find the addressing information of a specific
3106 appliance, and then directly accesses the appliance to control it.



3107

3108 **Figure 49. Interaction between OIC Client and OIC Server using another OIC Server**

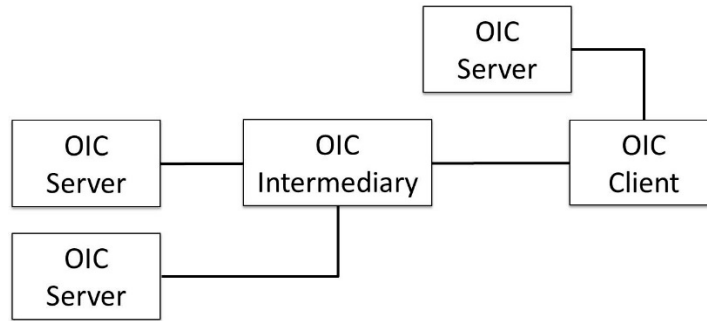
- 3109 • Interaction between OIC Client and OIC Server using OIC Intermediary (Figure 50). In this
3110 scenario an OIC Intermediary facilitates the interaction between the OIC Client and the OIC
3111 Server. A smartphone which controls appliances in a smart home via MQTT broker uses this
3112 scenario.



3113

3114 **Figure 50. Interaction between OIC Client and OIC Server using OIC Intermediary**

- 3115 • Interaction between OIC Client and OIC server using support from multiple OIC Servers and
3116 OIC intermediary (Figure 51). In this scenario, both OIC Server and OIC Intermediary roles are
3117 present to facilitate the transaction between the OIC Client and a specific OIC Server. An
3118 example scenario is when a smartphone first accesses a Resource Directory (RD) server to
3119 find the address to a specific appliance, then utilizes MQTT broker to deliver a command
3120 message to the appliance. The smartphone can utilize the mechanisms defined in CoRE
3121 Resource Directory such as default location, anycast address or DHCP (IETF draft-ietf-core-
3122 resource-directory-02) to discover the Resource Directory information.

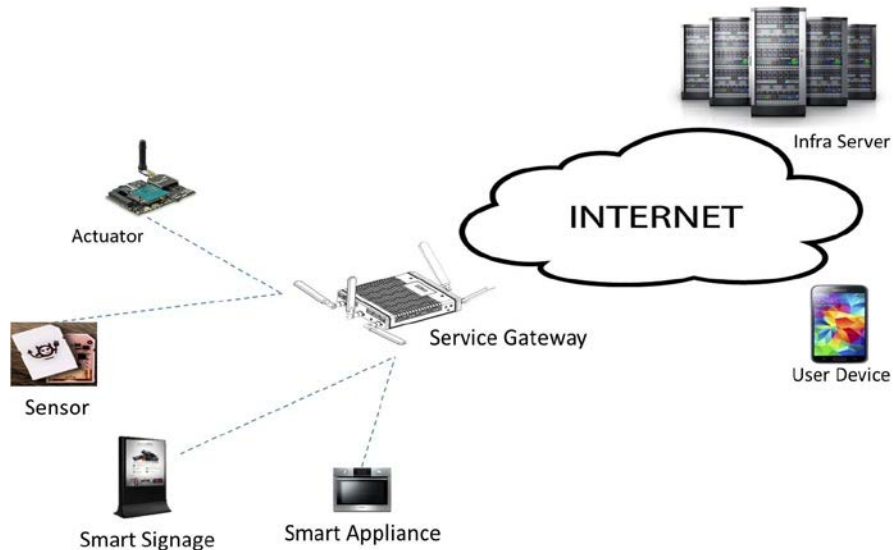


3123

3124 **Figure 51. Interaction between OIC Client and OIC Server using support from multiple OIC**
 3125 **Servers and OIC Intermediary**

3126 **B.2 OIC deployment model**

3127 In OIC Deployment, OIC Devices are deployed and interact via either wired or wireless connections.
 3128 OIC Devices are the physical entities that may host resources and play one or more OIC Roles.
 3129 There is no constraint on the structure of a deployment or number of OIC Devices in it. OIC
 3130 Architecture is flexible and scalable and capable of addressing large number of devices with
 3131 different device capabilities, including constrained devices which have limited memory and
 3132 capabilities. Constrained devices are defined and categorized in [TCNN].



3133

3134 **Figure 52. Example of OIC Devices**

3135 Figure 52 depicts a typical OIC deployment and set of OIC Devices, which may be divided in the
 3136 following categories:

- 3137 • **Things:** Networked devices which are able to interface with physical environments. Things are
 3138 the devices which are primarily controlled and monitored. Examples include smart appliances,
 3139 sensors, and actuators. Things mostly take the role of OIC Sever but they may also take the
 3140 role of OIC Client, for example in machine-to-machine communications.
- 3141 • **User Devices:** Devices employed by the users enabling the users to access resources and
 3142 services. Examples include smart phones, tablets, and wearable devices. User Devices mainly
 3143 take the role of OIC Client, but may also take the role of OIC Server or OIC Intermediary.

- 3144 • **Service Gateways:** Network equipment which take the role of OIC Intermediary. Examples are
3145 home gateways.
- 3146 • **Infra Servers:** Data centers residing in cloud infrastructure, which facilitate the interaction
3147 among OIC Devices by providing network services such as AAA, NAT traversal or discovery. It
3148 can also play the role of OIC Client or Intermediary

3149 **Annex C**
3150 (informative)

3151 **Other Resource Models and OIC Mapping**
3152

3153 **C.1 Multiple resource models**

3154 RESTful interactions are defined dependent on the resource model; hence, OIC Devices require a
3155 common understanding of the resource model for interoperability.

3156 There are multiple resource models defined by different organizations including OIC, IPSO Alliance
3157 and oneM2M, and used in the industry, which may restrict interoperability among respective
3158 ecosystems. The main differences from OIC resource model are as follows:

- 3159 • **Resource structure:** Resources may be defined to have properties (e.g., oneM2M defined
3160 resources), or may be defined as an atomic entity and not be decomposable into properties
3161 (e.g., IPSO alliance defined resources). For example, a smart light may be represented as a
3162 resource with an on-off property or a resource collection containing an on-off resource. In the
3163 former, on-off property doesn't have a URI of its own and can only be accessed indirectly via
3164 the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and
3165 can be directly manipulated.
- 3166 • **Resource name & type:** Resources may be allowed to be named freely and have their
3167 characteristics indicated using a resource type property (e.g., as defined in oneM2M).
3168 Alternatively, the name of resources may be defined a priori in a way that the name by itself is
3169 indicative of its characteristic (e.g., as defined by IPSO alliance). For example, in oneM2M
3170 resource model, a smart light can be named with no restrictions, such as 'LivingRoomLight_1"
3171 but in IPSO alliance resource model it is required to have the fixed Object name with numerical
3172 Object ID of "IPSO Light Control (3311)". Consequently, it's likely that in the former case the
3173 data path in URI is freely defined and in the latter case it is predetermined.
- 3174 • **Resource hierarchy:** Resources may be allowed to be organized in hierarchy where a resource
3175 contains another resource with a parent-child relationship (e.g., in oneM2M definition of
3176 resource model). Resources may also be required to have a flat structure and associate with
3177 other resources only by referencing their links.

3178 In addition to the above, different organizations use different syntax and define different features
3179 (e.g., resource interface), which preclude interoperability.

3180 **C.2 OIC approach for support of multiple resource models**

3181 In order to expand the IoT ecosystem the OIC Framework takes an inclusive approach for
3182 interworking with existing resource models. Specifically, the OIC Framework defines an OIC
3183 resource model while providing a mechanism to easily map to other models. By embracing existing
3184 resource models OIC is inclusive of existing ecosystems while allowing for the transition toward
3185 definition of a comprehensive resource model integrating all ecosystems.

3186 The following OIC characteristics enable support of other resource models:

- 3187 • **OIC resource model is the superset of multiple models:** the OIC resource model is defined
3188 as the superset of existing resource models. In other words, any existing resource model can
3189 be mapped to a subset of OIC resource model concepts.
- 3190 • **OIC Framework may allow for resource model negotiation:** the OIC Client and Server
3191 exchange the information about what resource model(s) each supports. Based on the
3192 exchanged information, the OIC Client and Server choose a resource model to perform RESTful
3193 interactions or to perform translation. This feature is out of scope of the current version of this
3194 specification, however, the following is a high level description for resource model negotiation..

3195 **C.3 Resource model indication**

3196 The OIC client and server exchange the information about what resource model(s) each supports.
3197 Based on the exchanged information, the OIC Client and Server choose a resource model to
3198 perform RESTful interactions or to perform translation. The exchange could be part of discovery
3199 and negotiation. Based on the exchange, the OIC Client and Server follow a procedure to ensure
3200 interoperability among them. They may choose a common resource model or execute translation
3201 between resource models.

- 3202 • **Resource model schema exchange:** The OIC Client and Server may share the resource
3203 model information when they initiate a RESTful interaction. They may exchange the information
3204 about which resource model they support as part of session establishment procedures.
3205 Alternatively, each request or response message may carry the indication of which resource
3206 model it is using. For example, [COAP] defines “Content-Format option” to indicate the
3207 “representation format” such as “application/json”. It’s possible to extend the Content-Format
3208 Option to indicate the resource model used with the representation format such as
3209 “application/ipsoson”.
- 3210 • **Ensuing procedures:** After the OIC Client and Server exchange the resource model
3211 information, they perform a suitable procedure to ensure interoperability among them. The
3212 simplest way is to choose a resource model supported by both the OIC Client and Server. In
3213 case there is no common resource model, the OIC Client and Server may interact through a
3214 3rd party.

3215 In addition to translation which can be resource intensive, a method based on profiles can be used
3216 in which an OIC implementation can accommodate multiple profiles and hence multiple
3217 ecosystems.

- 3218 • **Resource Model Profile:** the OIC Framework defines resource model profiles and
3219 implementers or users choose the active profile. The chosen profile constraints the OIC device
3220 to strict rules in how resources are defined, instantiated and interacted with. This would allow
3221 for interoperation with devices from the ecosystem identified by the profile (e.g., IPSO,
3222 OneM2M etc.). Although this enables an OIC Device to participate in and be part of any given
3223 ecosystem, this scheme does not allow for generic interoperability at runtime. While this
3224 approach may be suitable for resource constrained devices, more resource capable devices
3225 are expected to support more than one profile.

3226 **C.4 An Example Profile (IPSO profile)**

3227 IPSO defines smart objects that have specific resources and they take values determined by the
3228 data type of that resource. The smart object specification defines a category of such objects. Each
3229 resource represents a characteristic of the smart object being modelled.

3230 While the terms may be different, there are equivalent concepts in OIC to represent these terms.
3231 This section provides the equivalent OIC terms and then frames the IPSO smart object in OIC
3232 terms.

3233 The IPSO object Light Control defined in Section 16 of the IPSO Smart Objects 1.0 is used as the
3234 reference example.

3235 **C.4.1 Conceptual equivalence**

3236 The IPSO smart object definition is equivalent to an OIC Resource Type definition which defines
3237 the relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent
3238 to an OIC Property that like an IPSO Resource has a defined data type, enumeration of acceptable
3239 values, units, a general description and access modes (based on the OIC Interface).

3240 The general method for developing the equivalent OIC Resource Type from an IPSO Smart Object
 3241 definition is to ignore the Object ID and replace the Object URN with and OIC '.' (dot) separated
 3242 name that incorporates the IPSO object. Alternatively the Object URN can be used as the Resource
 3243 Type ID as is (as long as the URN does not contain any '.' (dots)) – using the same Object URN
 3244 as the Resource Type ID allows for compatibility when interacting with an IPSO compliant device.
 3245 The object URN based naming does not have any bearing for OIC to OIC interoperability and so
 3246 the OIC format is preferred – for OIC to OIC interoperability only the data model consistency is
 3247 required.

3248 Two models are available to render IPSO objects into OIC.

3249 1) One is where the IPSO Smart Object represents an OIC Resource. In this case, the IP Smart
 3250 Object is regarded as a resource with the Resource Type matching the description of the Smart
 3251 Object. Furthermore, each resource in the IPSO definition is represented as an Property in the
 3252 OIC Resource Type (the IPSO Resource ID is replaced with a string representing the OIC
 3253 Property). This is the preferred approach when the IPSO Data Model is expressed in the OIC
 3254 Resource Model.

3255 2) The other approach is to model an IPSO Smart Object as an OIC Collection. Each IPSO
 3256 Resource is then modelled as an OIC Resource with an OIC Resource Type that matches the
 3257 definition of the IPSO Resource. Each of these resource instances are then bound to the OIC
 3258 Collection that represents this IPSO Smart Object.

3259

3260 Below is an example showing how an IPSO LightControl Object is modelled as an OIC Resource.

3261 **OIC Resource Type: Light Control**

3262 Description: This Object is used to control a light source, such as a LED or other light. It allows a
 3263 light to be turned on or off and its dimmer setting to be controlled as a percentage value between
 3264 0 and 100. An optional colour setting enables a string to be used to indicate the desired colour.
 3265 Table 39 and Table 40 define the resource type and its properties, respectively.

3266 **Table 39. Light control resource type definition**

OIC 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

3267

3268 **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

3269

3270

3271 **Annex D**
3272 (informative)

3273 **Resource type definitions**
3274

3275 **D.1 List of resource type definitions**
3276

3277
3278 **D.2 OIC Configuration**

3279 **D.2.1 Introduction**

3280 Known resource that is hosted by every OIC Server. Allows for device specific information to be
3281 configured.

3282 **D.2.2 Wellknown URI**

3283 /oic/con

3284 **D.2.3 Resource Type**

3285 The resource type (rt) is defined as: oic.wk.con.

3286 **D.2.4 RAML Definition**

3287 `##RAML 0.8`

3288 `title: OIC Configuration`
3289 `version: v1-20150807`

3290 `traits:`

3291 `- interface`
3292 `queryParams:`

3293 `if:`
3294 `enum: ["oic.if.rw"]`

3295
3296 `/oic/con:`

3297 `description: |`
3298 `Known resource that is hosted by every OIC Server.`
3299 `Allows for device specific information to be configured.`
3300

3301 `is : ['interface']`

3302 `get:`

3303 `description: |`
3304 `Retrieves the current configuration settings`
3305

3306 `responses:`

3307 `200:`

3308 `body:`
3309 `application/json:`

3310 `schema: |`

3311 `{`
3312 `"id": "http://openinterconnect.org/oic.wk.con#",`
3313 `"$schema": "http://json-schema.org/draft-04/schema#",`
3314 `"definitions": {`
3315 `"oic.wk.con": {`
3316 `"type": "object",`
3317 `"properties": {`
3318 `"n": {`


```

3319         "type": "string",
3320         "description": "Human friendly name"
3321     },
3322     "loc": {
3323         "type": "string",
3324         "description": "Location information",
3325         "format": "json"
3326     },
3327     "locn": {
3328         "type": "string",
3329         "description": "Human Friendly Name"
3330     },
3331     "c": {
3332         "type": "string",
3333         "description": "Currency"
3334     },
3335     "r": {
3336         "type": "string",
3337         "description": "Region"
3338     }
3339 }
3340 }
3341 },
3342 "type": "object",
3343 "allOf": [
3344     { "$ref": "oic.core.json#/definitions/oic.core" },
3345     { "$ref": "#/definitions/oic.wk.con" }
3346 ],
3347 "required": [ "n" ]
3348 }
3349

```

```

3350 example: |
3351 {
3352     "rt": "oic.wk.con",
3353     "n": "My Friendly Device Name",
3354     "loc": "My Location Information",
3355     "locn": "My Location Name",
3356     "c": "USD",
3357     "r": "MyRegion"
3358 }
3359

```

3360 **post:**

```

3361 description: |
3362     Update the information about the OIC device
3363

```

```

3364 body:
3365     application/json

```

```

3366 schema: |
3367 {
3368     "id": "http://openinterconnect.org/oic.wk.con#",
3369     "$schema": "http://json-schema.org/draft-04/schema#",
3370     "definitions": {
3371         "oic.wk.con": {
3372             "type": "object",
3373             "properties": {
3374                 "n": {
3375                     "type": "string",
3376                     "description": "Human friendly name"
3377                 },
3378                 "loc": {
3379                     "type": "string",
3380                     "description": "Location information",
3381                     "format": "json"
3382                 },
3383                 "locn": {
3384                     "type": "string",
3385                     "description": "Human Friendly Name"

```

```

3386         },
3387         "c": {
3388             "type": "string",
3389             "description": "Currency"
3390         },
3391         "r": {
3392             "type": "string",
3393             "description": "Region"
3394         }
3395     }
3396 },
3397 },
3398 "type": "object",
3399 "allof": [
3400     { "$ref": "oic.core.json#/definitions/oic.core" },
3401     { "$ref": "#/definitions/oic.wk.con" }
3402 ],
3403 "required": [ "n" ]
3404 }
3405
3406 example: |
3407 {
3408     "n": "My Friendly Device Name"
3409 }
3410
3411 responses:
3412 200:
3413     body:
3414         application/json:
3415             schema: |
3416                 {
3417                     "id": "http://openinterconnect.org/oic.wk.con#",
3418                     "$schema": "http://json-schema.org/draft-04/schema#",
3419                     "definitions": {
3420                         "oic.wk.con": {
3421                             "type": "object",
3422                             "properties": {
3423                                 "n": {
3424                                     "type": "string",
3425                                     "description": "Human friendly name"
3426                                 },
3427                                 "loc": {
3428                                     "type": "string",
3429                                     "description": "Location information",
3430                                     "format": "json"
3431                                 },
3432                                 "locn": {
3433                                     "type": "string",
3434                                     "description": "Human Friendly Name"
3435                                 },
3436                                 "c": {
3437                                     "type": "string",
3438                                     "description": "Currency"
3439                                 },
3440                                 "r": {
3441                                     "type": "string",
3442                                     "description": "Region"
3443                                 }
3444                             }
3445                         }
3446                     },
3447                     "type": "object",
3448                     "allof": [
3449                         { "$ref": "oic.core.json#/definitions/oic.core" },
3450                         { "$ref": "#/definitions/oic.wk.con" }
3451                     ],
3452                     "required": [ "n" ]

```

```

3453     }
3454
3455     example: |
3456     {
3457         "n": "My Friendly Device Name"
3458     }
3459

```

3460 D.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
---------------	------------	-----------	-------------	-------------

3461 D.2.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/con		get	post		

3462 D.3 OIC Logical Device

3463 D.3.1 Introduction

3464 Known resource that is hosted by every OIC Server. Allows for logical device specific information
 3465 to be discovered.

3466 D.3.2 Wellknown URI

3467 /oic/d

3468 D.3.3 Resource Type

3469 The resource type (rt) is defined as: oic.wk.d.

3470 D.3.4 RAML Definition

```

3471 #%RAML 0.8
3472 title: OIC Root Device
3473 version: v1-20150811
3474 traits:
3475   - interface
3476     queryParameters:
3477       if:
3478         enum: ["oic.if.r"]
3479
3480 /oic/d:
3481   description: |
3482     Known resource that is hosted by every OIC Server.
3483     Allows for logical device specific information to be discovered.
3484
3485   is : ['interface']
3486   get:
3487     description: |
3488       Retrieve the information about the OIC device
3489
3490   responses:
3491     200:
3492       body:
3493         application/json:
3494           schema: |
3495             {
3496               "$schema": "http://json-schemas.org/draft-04/schema#",
3497               "id": "http://openinterconnect.org/oic.wk.d#",
3498               "definitions": {

```

```

3499         "oic.wk.d": {
3500             "type": "object",
3501             "properties": {
3502                 "n": {
3503                     "type": "string",
3504                     "description": "ReadOnly, Human friendly name"
3505                 },
3506                 "di": {
3507                     "type": "string",
3508                     "description": "ReadOnly, Unique identifier for device (UUID)",
3509                     "format": "uuid"
3510                 },
3511                 "icv": {
3512                     "type": "string",
3513                     "description": "ReadOnly, The version of the OIC Server"
3514                 },
3515                 "dmv": {
3516                     "type": "string",
3517                     "description": "ReadOnly, The spec version of the vertical specification",
3518                     "format": "csv"
3519                 }
3520             }
3521         },
3522     },
3523     "type": "object",
3524     "allOf": [
3525         { "$ref": "#/definitions/oic.wk.d" }
3526     ],
3527     "required": [ "n", "di", "icv" ]
3528 }
3529
3530 example: |
3531 {
3532     "n":      "Device 1",
3533     "rt":    "oic.wk.d",
3534     "di":    "54919CA5-4101-4AE4-595B-353C51AA983C",
3535     "icv":   "OIC 1.0"
3536 }
3537

```

3538 D.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string	yes	Read Write	ReadOnly, Human friendly name
di	string	yes	Read Only	Unique Identifier For Device (Uuid)
icv	string	yes	Read Only	The Version Of The Oic Server
dmv	string		Read Only	The Spec Version Of The Vertical Specification

3539 D.3.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/d		get			

3540 D.4 OIC Interface Types

3541 D.4.1 Introduction

3542 List of resource interfaces that are supported by this OIC Server

3543 D.4.2 Wellknown URI

3544 /oic/ifs

3545 D.4.3 Resource Type

3546 The resource type (rt) is defined as: oic.wk.ifs.

3547 **D.4.4 RAML Definition**

```

3548 #%RAML 0.8
3549 title: OIC Interface Types
3550 version: v1-20150811
3551 traits:
3552   - interface
3553     queryParameters:
3554       if:
3555         enum: ["oic.if.r"]
3556
3557 /oic/ifs:
3558   description: |
3559     List of resource interfaces that are supported by this OIC Server
3560
3561   is : ['interface']
3562   get:
3563     description: |
3564       Retrieve the resource interface list
3565
3566   responses:
3567     200:
3568       body:
3569         application/json:
3570           schema: |
3571             {
3572               "$schema": "http://json-schemas.org/draft-04/schema#",
3573               "id": "http://openinterconnect.org/oic.wk.ifs#",
3574               "definitions": {
3575                 "oic.wk.ifs": {
3576                   "type": "object",
3577                   "properties": {
3578                     "il": {
3579                       "type": "string",
3580                       "description": "Readonly, list of interface names",
3581                       "format": "bsv"
3582                     }
3583                   }
3584                 },
3585               },
3586               "type": "object",
3587               "allOf": [
3588                 { "$ref": "#/definitions/oic.wk.ifs" }
3589               ],
3590               "required": ["il"]
3591             }
3592
3593           example: |
3594             {
3595               "rt": "oic.wk.ifs",
3596               "il": "oic.if.ll oic.if.bat oic.if.r"
3597             }
3598

```

3599 **D.4.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
il	string	yes	Read Write	Readonly, list of interface names

3600 **D.4.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
----------	--------	------	--------	--------	--------

/oic/ifs		get			
----------	--	-----	--	--	--

3601 D.5 OIC Maintenance

3602 D.5.1 Introduction

3603 The resource through which an OIC Device is maintained and can be used for diagnostic purposes.
 3604 fr (Factory Reset) is a boolean. The value 0 means No action (Default), the value 1 means Start
 3605 Factory Reset After factory reset, this value shall be changed back to the default value rb (Reboot)
 3606 is a boolean. The value 0 means No action (Default), the value 1 means Start Reboot After Reboot,
 3607 this value shall be changed back to the default value ssc (Start Stat Collection) is a boolean. The
 3608 value 0 means No collection of statistics, the value 1 means Starts collecting statistics

3609 D.5.2 Wellknown URI

3610 /oic/mnt

3611 D.5.3 Resource Type

3612 The resource type (rt) is defined as: oic.wk.mnt.

3613 D.5.4 RAML Definition

```

3614 #%RAML 0.8
3615 title: OIC Maintenance
3616 version: v1-20150811
3617 traits:
3618   - interface
3619     queryParameters:
3620       if:
3621         enum: ["oic.if.r"]
3622
3623 /oic/mnt:
3624   description: |
3625     The resource through which an OIC Device is maintained and can be used for diagnostic purposes.
3626     fr (Factory Reset) is a boolean.
3627     The value 0 means No action (Default), the value 1 means Start Factory Reset
3628     After factory reset, this value shall be changed back to the default value
3629     rb (Reboot) is a boolean.
3630     The value 0 means No action (Default), the value 1 means Start Reboot
3631     After Reboot, this value shall be changed back to the default value
3632     ssc (Start Stat Collection) is a boolean.
3633     The value 0 means No collection of statistics, the value 1 means Starts collecting statistics
3634
3635   is : ['interface']
3636   get:
3637     description: |
3638       Retrieve the maintenance action status
3639
3640     queryParameters:
3641       if:
3642         enum: oic.if.r
3643     responses:
3644       200:
3645         body:
3646           application/json:
3647             schema: |
3648               {
3649                 "$schema": "http://json-schemas.org/draft-04/schema#",
3650                 "id": "http://openinterconnect.org/oic.wk.mnt#",
3651                 "definitions": {
  
```

```

3652         "oic.wk.mnt": {
3653             "type": "object",
3654             "fr": {
3655                 "type": "boolean",
3656                 "description": "Factory Reset"
3657             },
3658             "rb": {
3659                 "type": "boolean",
3660                 "description": "Reboot Action"
3661             },
3662             "ssc": {
3663                 "type": "boolean",
3664                 "description": "Start Stat Collection Action Toggle"
3665             }
3666         },
3667     },
3668     "type": "object",
3669     "allOf": [
3670         {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},
3671         {"$ref": "#/definitions/oic.wk.mnt" }
3672     ],
3673     "required": ["fr", "rb", "ssc"]
3674 }
3675
3676 example: |
3677 {
3678     "rt":    "oic.wk.mnt",
3679     "n":     "My Maintenance Actions",
3680     "fr":    false,
3681     "rb":    false,
3682     "ssc":   false
3683 }
3684
3685 post:
3686     description: |
3687         Set the maintenance action(s)
3688
3689     queryParameters:
3690         if:
3691             enum: oic.if.rw
3692
3693     body:
3694         application/json
3695
3696     schema: |
3697     {
3698         "$schema": "http://json-schemas.org/draft-04/schema#",
3699         "id": "http://openinterconnect.org/oic.wk.mnt#",
3700         "definitions": {
3701             "oic.wk.mnt": {
3702                 "type": "object",
3703                 "fr": {
3704                     "type": "boolean",
3705                     "description": "Factory Reset"
3706                 },
3707                 "rb": {
3708                     "type": "boolean",
3709                     "description": "Reboot Action"
3710                 },
3711                 "ssc": {
3712                     "type": "boolean",
3713                     "description": "Start Stat Collection Action Toggle"
3714                 }
3715             }
3716         },
3717         "type": "object",
3718         "allOf": [
3719             {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},

```

```

3718         {"$ref": "#/definitions/oic.wk.mnt" }
3719     ],
3720     "required": ["fr", "rb", "ssc"]
3721 }
3722
3723     example: |
3724     {
3725         "n":     "My Maintenance Actions",
3726         "fr":    false,
3727         "rb":    false,
3728         "ssc":   true
3729     }
3730
3731     responses:
3732     200:
3733         body:
3734         application/json:
3735             schema: |
3736             {
3737                 "$schema": "http://json-schemas.org/draft-04/schema#",
3738                 "id": "http://openinterconnect.org/oic.wk.mnt#",
3739                 "definitions": {
3740                     "oic.wk.mnt": {
3741                         "type": "object",
3742                         "fr": {
3743                             "type": "boolean",
3744                             "description": "Factory Reset"
3745                         },
3746                         "rb": {
3747                             "type": "boolean",
3748                             "description": "Reboot Action"
3749                         },
3750                         "ssc": {
3751                             "type": "boolean",
3752                             "description": "Start Stat Collection Action Toggle"
3753                         }
3754                     }
3755                 },
3756                 "type": "object",
3757                 "allOf": [
3758                     {"$ref": "http://openinterconnect.org/oic.core.json#/definitions/oic.core"},
3759                     {"$ref": "#/definitions/oic.wk.mnt" }
3760                 ],
3761                 "required": ["fr", "rb", "ssc"]
3762             }
3763
3764             example: |
3765             {
3766                 "n":     "My Maintenance Actions",
3767                 "fr":    false,
3768                 "rb":    false,
3769                 "ssc":   true
3770             }
3771

```

3772 D.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
fr	boolean	yes	Read Write	Factory Reset
rb	boolean	yes	Read Write	Reboot Action
ssc	boolean	yes	Read Write	Start Stat Collection Action Toggle

3773 **D.5.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/oic/mnt		get	post		

3774 **D.6 OIC Monitoring**

3775 **D.6.1 Introduction**

3776 The resource through which an OIC Device is monitored.

3777 **D.6.2 Wellknown URI**

3778 /oic/mon

3779 **D.6.3 Resource Type**

3780 The resource type (rt) is defined as: oic.wk.mon.

3781 **D.6.4 RAML Definition**

```
3782 #%RAML 0.8
3783 title: OIC Monitoring
3784 version: v1-20150401
3785 traits:
3786   - interface
3787     queryParameters:
3788       if:
3789         enum: ["oic.if.r"]
3790
3791 /oic/mon:
3792   description: |
3793     The resource through which an OIC Device is monitored.
3794
3795   is : ['interface']
3796   get:
3797     description: |
3798       Retrieve the monitor information
3799
3800   responses:
3801     200:
3802       body:
3803         application/json:
3804           schema: |
3805             {
3806               "$schema": "http://json-schemas.org/draft-04/schema#",
3807               "id": "http://openinterconnect.org/oic.wk.mon#",
3808               "definitions": {
3809                 "oic.wk.mon": {
3810                   "type": "object",
3811                   "properties": {
3812                     "av": {
3813                       "type": "boolean",
3814                       "description": "ReadOnly, Indicates if the device is available or not on
3815 the network (like ping)",
3816                     },
3817                     "lat": {
3818                       "type": "integer",
3819                       "description": "ReadOnly, Indicates the elapsed time in seconds after the
3820 device was invoked or acted upon"
3821                     },
3822                     "ds": {
3823                       "type": "string",
```

```

3824         "description": "ReadOnly, Contains Device Statistics Info (no. of received
3825 packets, no. of sent packets, time to respond etc.)",
3826         "format": "csv"
3827     }
3828 }
3829 }
3830 },
3831 "type": "object",
3832 "allOf": [
3833     {"$ref": "oic.core.json#/definitions/oic.core"},
3834     {"$ref": "#/definitions/oic.wk.mon"}
3835 ],
3836 "required": ["av", "lat"]
3837 }
3838
3839 example: |
3840 {
3841     "rt": "oic.wk.mon",
3842     "name": "My Monitor Information",
3843     "av": true,
3844     "lat": 50,
3845     "ds": "1500, 2750, 0"
3846 }
3847

```

3848 D.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
av	boolean	yes	Read Only	Indicates If The Device Is Available Or Not On The Network (Like Ping)
lat	integer	yes	Read Only	Indicates The Elapsed Time In Seconds After The Device Was Invoked Or Acted Upon
ds	string		Read Only	Contains Device Statistics Info (No. Of Received Packets, No. Of Sent Packets, Time To Respond Etc.)

3849 D.6.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/mon		get			

3850 D.7 OIC Base Platform

3851 D.7.1 Introduction

3852 Known resource that is defines the platform on which an OIC Server is hosted. Allows for platform
3853 specific information to be discovered.

3854 D.7.2 Wellknown URI

3855 /oic/p

3856 D.7.3 Resource Type

3857 The resource type (rt) is defined as: oic.wk.p.

3858 D.7.4 RAML Definition

```

3859 #%RAML 0.8
3860 title: OIC Base Platform
3861 version: v1-20150804
3862 traits:
3863   - interface
3864     queryParameters:

```

```

3865         if:
3866             enum: ["oic.if.r"]
3867
3868 /oic/p:
3869     description: |
3870         Known resource that is defines the platform on which an OIC Server is hosted.
3871         Allows for platform specific information to be discovered.
3872
3873     is : ['interface']
3874     get:
3875         description: |
3876             Retrieve the information about the OIC Platform
3877
3878     responses:
3879         200:
3880             body:
3881                 application/json:
3882                     schema: |
3883                         {
3884                             "$schema": "http://json-schemas.org/draft-04/schema#",
3885                             "id": "http://openinterconnect.org/oic.wk.p#",
3886                             "definitions": {
3887                                 "oic.wk.p": {
3888                                     "type": "object",
3889                                     "properties": {
3890                                         "pi": {
3891                                             "type": "string",
3892                                             "description": "ReadOnly, Platform Identifier"
3893                                         },
3894                                         "mnmn": {
3895                                             "type": "string",
3896                                             "description": "ReadOnly, Manufacturer Name",
3897                                             "maxLength": 16
3898                                         },
3899                                         "mnml": {
3900                                             "type": "string",
3901                                             "description": "ReadOnly, Manufacturer's URL",
3902                                             "maxLength": 32,
3903                                             "format": "uri"
3904                                         },
3905                                         "mnmo": {
3906                                             "type": "string",
3907                                             "description": "ReadOnly, Model number as designated by manufacturer"
3908                                         },
3909                                         "mndt": {
3910                                             "type": "string",
3911                                             "description": "ReadOnly, Manufacturing Date",
3912                                             "format": "date-time"
3913                                         },
3914                                         "mnpv": {
3915                                             "type": "string",
3916                                             "description": "ReadOnly, Platform Version"
3917                                         },
3918                                         "mnos": {
3919                                             "type": "string",
3920                                             "description": "ReadOnly, Platform Resident OS Version"
3921                                         },
3922                                         "mnhw": {
3923                                             "type": "string",
3924                                             "description": "ReadOnly, Platform Hardware Version"
3925                                         },
3926                                         "mnfv": {
3927                                             "type": "string",
3928                                             "description": "ReadOnly, Manufacturer's firmware version"
3929                                         },

```

```

3930         "mns1": {
3931             "type": "string",
3932             "description": "ReadOnly, Manufacturer's Support Information URL",
3933             "format": "uri"
3934         },
3935         "st": {
3936             "type": "string",
3937             "description": "ReadOnly, Reference time for the device",
3938             "format": "date-time"
3939         }
3940     }
3941 },
3942 },
3943 "type": "object",
3944 "allOf": [
3945     { "$ref": "#/definitions/oic.wk.p" }
3946 ],
3947 "required": [ "pi", "mnmn" ]
3948 }
3949

```

```

3950 example: |
3951 {
3952     "pi": "my-platform-identfier",
3953     "rt": "oic.wk.p",
3954     "mnmn": "Acme, Inc"
3955 }
3956

```

3957 D.7.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
pi	string	yes	Read Only	Platform Identifier
mnmn	string	yes	Read Only	Manufacturer Name
maxLength				
mnml	string		Read Only	Manufacturer'S Url
maxLength				
mnmo	string		Read Only	Model Number As Designated By Manufacturer
mndt	string		Read Only	Manufacturing Date
mpv	string		Read Only	Platform Version
mnos	string		Read Write	Readonly, Platform Resident OS Version
mnhw	string		Read Write	Readonly, Platform Hardware Version
mnfv	string		Read Only	Manufacturer'S Firmware Version
mns1	string		Read Only	Manufacturer'S Support Information Url
st	string		Read Only	Reference Time For The Device

3958 D.7.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/p		get			

3959 D.8 OIC Ping

3960 D.8.1 Introduction

3961 The resource using which an OIC Client keeps its Connection with an OIC Server active.

3962 D.8.2 Wellknown URI

3963 /oic/ping

3964 **D.8.3 Resource Type**

3965 The resource type (rt) is defined as: oic.wk.ping.

3966 **D.8.4 RAML Definition**

```
3967 #%RAML 0.8
3968 title: OIC Ping
3969 version: v1-20150814
3970 traits:
3971   - interface
3972     queryParameters:
3973       if:
3974         enum: ["oic.if.rw"]
3975
3976 /oic/ping:
3977   description: |
3978     The resource using which an OIC Client keeps its Connection with an OIC Server active.
3979
3980   is : ['interface']
3981   get:
3982     description: |
3983       Retrieve the ping information
3984
3985     responses:
3986       200:
3987         body:
3988           application/json:
3989             schema: |
3990               {
3991                 "$schema": "http://json-schemas.org/draft-04/schema#",
3992                 "id": "http://openinterconnect.org/oic.wk.ping#",
3993                 "definitions": {
3994                   "oic.wk.ping": {
3995                     "type": "object",
3996                     "properties": {
3997                       "in": {
3998                         "type": "integer",
3999                         "description": "ReadWrite, Indicates the interval for which connection
4000 shall be kept alive"
4001                       }
4002                     }
4003                   }
4004                 },
4005                 "type": "object",
4006                 "allOf": [
4007                   {
4008                     "$ref": "oic.core.json#/definitions/oic.core"
4009                   },
4010                   {
4011                     "$ref": "#/definitions/oic.wk.ping"
4012                   }
4013                 ],
4014                 "required": [
4015                   "in"
4016                 ]
4017               }
4018
4019           example: |
4020             {
4021               "rt": "oic.wk.ping",
4022               "name": "Ping Information",
```

```

4023         "in": 16
4024     }
4025

```

4026 D.8.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
in	integer		Read Write	ReadWrite, Indicates the interval for which connection shall be kept alive
in				

4027 D.8.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/ping		get			

4028 D.9 OIC Discoverable Resources

4029 D.9.1 Introduction

4030 The resource through which the corresponding OIC Server is discovered and introspected for
4031 available resources.

4032 D.9.2 Wellknown URI

4033 /oic/res

4034 D.9.3 Resource Type

4035 The resource type (rt) is defined as: oic.wk.res.

4036 D.9.4 RAML Definition

```

4037 #%RAML 0.8
4038 title: OIC Discoverable Resources
4039 version: v1-20150807
4040 traits:
4041   - interface
4042     queryParameters:
4043       if:
4044         enum: ["oic.if.ll"]
4045
4046 /oic/res:
4047   description: |
4048     The resource through which the corresponding OIC Server is discovered and introspected for
4049     available resources.
4050
4051   is : ['interface']
4052   get:
4053     description: |
4054       Retrieve the discoverable resource set
4055
4056   responses:
4057     200:
4058       body:
4059         application/json:
4060           schema: |
4061             {
4062               "$schema": "http://json-schema.org/draft-v4/schema#",
4063               "id": "http://openinterconnect.org/schemas/oic.wk.res.json/",
4064               "definitions": {

```

```

4065         "oic.res-links.json": {
4066             "type": "object",
4067             "properties": {
4068                 "n": {
4069                     "type": "string",
4070                     "description": "ReadOnly, Human friendly name"
4071                 },
4072                 "di": {
4073                     "description": "The device identifier as indicated by the /oic/d resource
4074 of the device",
4075                     "type": "string",
4076                     "format": "UUID"
4077                 },
4078                 "mpro": {
4079                     "description": "ReadOnly, Supported messaging protocols",
4080                     "type": "string"
4081                 },
4082                 "links": {
4083                     "type": "array",
4084                     "items": {
4085                         "$ref": "oic.web-link.json#"
4086                     }
4087                 }
4088             }
4089         },
4090     },
4091     "description": "The list of resources expressed as web links",
4092     "type": "array",
4093     "items": {
4094         "$ref": "#/definitions/oic.res-links.json"
4095     },
4096     "required": ["di", "links"]
4097 }
4098

```

```

4099     example: |
4100         [{
4101             "rt": "oic.wk.res",
4102             "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
4103             "links":
4104                 [
4105                     {
4106                         "href": "/res",
4107                         "rel": "self",
4108                         "rt": "oic.r.collection",
4109                         "if": "oic.if.ll" },
4110                     {
4111                         "href": "/smartDevice",
4112                         "rel": "contained",
4113                         "rt": "oic.d.smartDevice",
4114                         "if": "oic.if.a"
4115                     }
4116                 ]
4117         }]
4118

```

4119 D.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Only	Human Friendly Name
di		yes		
mpro				
links	array	yes		
items				
items				

4120 D.9.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
----------	--------	------	--------	--------	--------

/oic/res		get			
----------	--	-----	--	--	--

4121 D.10 OIC Resource Types

4122 D.10.1 Introduction

4123 List of resource types that are supported by this OIC Server

4124 D.10.2 Wellknown URI

4125 /oic/rts

4126 D.10.3 Resource Type

4127 The resource type (rt) is defined as: oic.wk.rts.

4128 D.10.4 RAML Definition

4129 `##RAML 0.8`

4130 `title: OIC Resource Types`

4131 `version: v1-20150811`

4132 `traits:`

4133 `- interface`

4134 `queryParameters:`

4135 `if:`

4136 `enum: ["oic.if.r"]`

4137

4138 `/oic/rts:`

4139 `description: |`

4140 `List of resource types that are supported by this OIC Server`

4141

4142 `is : ['interface']`

4143 `get:`

4144 `description: |`

4145 `Retrieve the resource type list`

4146

4147 `responses:`

4148 `200:`

4149 `body:`

4150 `application/json:`

4151 `schema: |`

```

4152           {
4153             "$schema": "http://json-schemas.org/draft-04/schema#",
4154             "id": "http://openinterconnect.org/oic.wk.rts#",
4155             "definitions": {
4156               "oic.wk.rts": {
4157                 "type": "object",
4158                 "properties": {
4159                   "t1": {
4160                     "type": "string",
4161                     "description": "Readonly, list of resource type names",
4162                     "format": "bsv"
4163                   }
4164                 }
4165             }
4166           },
4167           "type": "object",
4168           "allOf": [
4169             { "$ref": "#/definitions/oic.wk.rts" }
4170           ],
4171           "required": ["t1"]
4172         }
4173
```



```

4174     example: |
4175         {
4176           "rt": "oic.wk.rts",
4177           "tl": "oic.r.example oic.r.other-example"
4178         }
4179

```

4180 D.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
tl	string	yes	Read Write	Readonly, list of resource type names

4181 D.10.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/oic/rts		get			

4182 D.11 Scenes (Top level)

4183 D.11.1 Introduction

4184 Toplevel Scene resource. This resource is a generic collection resource. The rts value shall contain
 4185 oic.sceneCollection resource types.

4186 D.11.2 Wellknown URI

4187 /SceneListResURI

4188 D.11.3 Resource Type

4189 The resource type (rt) is defined as: oic.wk.sceneList.

4190 D.11.4 RAML Definition

```

4191 #%RAML 0.8
4192 title: OICScene
4193 version: v1.0-20150630
4194 traits:
4195   - interface
4196     queryParameters:
4197       if:
4198         enum: ["oic.if.a", "oic.if.ll"]
4199
4200 /SceneListResURI:
4201   description: |
4202     Toplevel Scene resource.
4203     This resource is a generic collection resource.
4204     The rts value shall contain oic.sceneCollection resource types.
4205
4206   get:
4207     description: |
4208       Provides the current list of web links pointing to scenes
4209
4210     responses:
4211       200:
4212         body:
4213           application/json:
4214             schema: |
4215               {
4216                 "$schema": "http://json-schema.org/draft-04/schema#",
4217                 "id": "http://openinterconnect.org/schemas/oic.collection.json#",

```

```

4218     "title" : "Collection",
4219     "definitions": {
4220         "oic.collection": {
4221             "type": "object",
4222             "properties": {
4223                 "n": {
4224                     "type": "string",
4225                     "description": "Used to name the collection",
4226                     "format": "UTF8"
4227                 },
4228                 "id": {
4229                     "oneOf" : [
4230                         { "type": "number", "description": "if id
4231 property is an number" },
4232                         { "type": "string", "description": "if id
4233 property is an number" }
4234                     ]
4235                 },
4236                 "rts": {
4237                     "type": "string",
4238                     "description": "Defines the list of allowable
4239 resource types in links included in the collection; new links being created can only be from this
4240 list",
4241                     "format": "UTF8"
4242                 },
4243                 "links": {
4244                     "type": "array",
4245                     "description": "Array of OIC web links that are
4246 reference from this collection",
4247                     "items" : {
4248                         "allOf": [
4249                             { "$ref":
4250 "http://openinterconnect.org/schemas/oic.web-link.json#" },
4251                             { "required" : [ "ins" ] }
4252                         ]
4253                     }
4254                 },
4255                 "required": [ "links" ]
4256             }
4257         },
4258     },
4259     "type": "object",
4260     "allOf" : [
4261         { "$ref": "oic.core.json#/definitions/oic.core" },
4262         { "$ref": "#/definitions/oic.collection" }
4263     ]
4264 }
4265 }
4266
4267     example: |
4268     {
4269         "rt":      "oic.wk.sceneList",
4270         "n":       "list of scene Collections",
4271         "id":      "my_sceneList_1",
4272         "rts":     "oic.wk.sceneCollection",
4273         "links":  [
4274             ]
4275     }
4276
4277     post:
4278     description: |
4279     Provides the action to create a new sceneCollection in the SceneList resource
4280     The only resource type that is allowed to be created "oic.wk.sceneCollection".
4281     The example contains 3 scene values off, Reading and TVWatching.
4282
4283     body:
4284     application/json
4285
4286     schema: |

```

```

4286     {
4287         "$schema": "http://json-schema.org/draft-04/schema#",
4288         "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4289         "title": "Scene Collection",
4290         "definitions": {
4291             "oic.sceneCollection": {
4292                 "type": "object",
4293                 "properties": {
4294                     "lastScene": {
4295                         "type": "string",
4296                         "description": "Last selected Scene, shall be part of
sceneValues",
4297                         "format": "UTF8"
4298                     },
4299                     "sceneValues": {
4300                         "type": "string",
4301                         "description": "ReadOnly, All available scene
values",
4302                         "format": "CSV"
4303                     },
4304                     "n": {
4305                         "type": "string",
4306                         "description": "Used to name the Scene collection",
4307                         "format": "UTF8"
4308                     },
4309                     "id": {
4310                         "type": "string",
4311                         "description": "A unique string that could be a hash or
similarly unique"
4312                     },
4313                     "rts": {
4314                         "type": "string",
4315                         "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list",
4316                         "format": "UTF8"
4317                     },
4318                     "links": {
4319                         "type": "array",
4320                         "description": "Array of OIC web links that are
reference from this collection",
4321                         "items": {
4322                             "allOf": [
4323                                 { "$ref": "http://openinterconnect.org/schemas/oic.web-link.json#" },
4324                                 { "required": [ "ins" ] }
4325                             ]
4326                         }
4327                     },
4328                     "required": [ "lastScene", "sceneValues", "rts", "id" ]
4329                 }
4330             }
4331         },
4332         "type": "object",
4333         "allOf": [
4334             { "$ref": "oic.core.json#/definitions/oic.core" },
4335             { "$ref": "#/definitions/oic.sceneCollection" }
4336         ]
4337     }
4338
4339     example: |
4340     {
4341         "scenevalue": "off",
4342         "sceneValues": "off,Reading,TVWatching",
4343         "lastScene": "off",
4344         "rt": "oic.wk.sceneCollection",
4345         "n": "my first scene",
4346         "id": "my_scenel",
4347         "rts": "oic.r.sceneMember"

```

```

4356     }
4357
4358     responses:
4359         200:
4360             description: |
4361                 Indicates that the target resource was created.
4362                 The created resource attributes are provided in the response,
4363                 including the server generated identifier.
4364
4365             body:
4366                 application/json:
4367                     schema: |
4368                         {
4369                             "$schema": "http://json-schema.org/draft-04/schema#",
4370                             "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4371                             "title" : "Scene Collection",
4372                             "definitions": {
4373                                 "oic.sceneCollection": {
4374                                     "type": "object",
4375                                     "properties": {
4376                                         "lastScene": {
4377                                             "type": "string",
4378                                             "description": "Last selected Scene, shall be part of
4379 sceneValues",
4380                                             "format": "UTF8"
4381                                         },
4382                                         "sceneValues": {
4383                                             "type": "string",
4384                                             "description": "ReadOnly, All available scene
4385 values",
4386                                             "format": "CSV"
4387                                         },
4388                                         "n": {
4389                                             "type": "string",
4390                                             "description": "Used to name the Scene collection",
4391                                             "format": "UTF8"
4392                                         },
4393                                         "id": {
4394                                             "type": "string",
4395                                             "description" : "A unique string that could be a hash or
4396 similarly unique"
4397                                         },
4398                                         "rts": {
4399                                             "type": "string",
4400                                             "description": "ReadOnly, Defines the list of
4401 allowable resource types in links included in the collection; new links being created can only be
4402 from this list",
4403                                             "format": "UTF8"
4404                                         },
4405                                         "links": {
4406                                             "type": "array",
4407                                             "description": "Array of OIC web links that are
4408 reference from this collection",
4409                                             "items" : {
4410                                                 "allOf": [
4411                                                     { "$ref":
4412 "http://openinterconnect.org/schemas/oic.web-link.json#" },
4413                                                     { "required" : [ "ins" ] }
4414                                                 ]
4415                                             }
4416                                         }
4417                                     },
4418                                     "required": [ "lastScene", "sceneValues", "rts", "id" ]
4419                                 }
4420                             },
4421                             "type": "object",
4422                             "allOf" : [
4423

```

```

4424         { "$ref": "oic.core.json#/definitions/oic.core" },
4425         { "$ref": "#/definitions/oic.sceneCollection" }
4426     ]
4427 }
4428
4429     example: |
4430     {
4431         "scenevalue": "off",
4432         "sceneValues": "off,Reading,TVWatching",
4433         "lastScene": "off",
4434         "rt":         "oic.r.sceneCollection",
4435         "n":         "mymembername",
4436         "link":      "coap://newsScene",
4437         "id":        "0685B960-736F-46F7-BEC0-9E6CBD671ADC1",
4438         "rts":       "oic.r.sceneMember"
4439     }
4440
4441     delete:
4442         description: |
4443             No change from collection.
4444             When delete is used with the URI of the collection without and query parameters then the
4445             entire collection is deleted
4446             When the delete uses the "ins" parameter with the value of a specific link then only that
4447             link is deleted
4448
4449         queryParameters:
4450             ins:
4451                 type: string
4452                 description: Delete the Web link identified by the string - could be a UUID.
4453
4454             required: false
4455             example: DELETE /mycollection?ins="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"
4456
4457         responses:
4458             200:
4459                 description: |
4460                     The web link instance or the the entire collection has been successfully deleted
4461
4462             400:
4463                 description: |
4464                     The request is invalid
4465
4466

```

D.11.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the collection
id	object			
rts	string		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
links	array	yes	Read Write	Array of OIC web links that are reference from this collection
items				

D.11.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
----------	--------	------	--------	--------	--------

/SceneListResURI		get	post	delete	
------------------	--	-----	------	--------	--

4468 D.12 Scene Collections

4469 D.12.1 Introduction

4470 Collection that models a set of Scenes. This resource is a generic collection resource with
 4471 additional parameters. The rts value shall contain oic.sceneMember resource types. The additional
 4472 parameters are lastScene, this is the scene value last set by any OIC Client sceneValueList,
 4473 this is the list of available scenes lastScene shall be listed in sceneValueList.

4474 D.12.2 Wellknown URI

4475 /SceneCollectionResURI

4476 D.12.3 Resource Type

4477 The resource type (rt) is defined as: oic.wk.sceneCollection.

4478 D.12.4 RAML Definition

```

4479 #%RAML 0.8
4480 title: OICScene
4481 version: v1.0-20150630
4482 traits:
4483   - interface
4484     queryParameters:
4485       if:
4486         enum: ["oic.if.a", "oic.if.ll"]
4487
4488 /SceneCollectionResURI:
4489   description: |
4490     Collection that models a set of Scenes.
4491     This resource is a generic collection resource with additional parameters.
4492     The rts value shall contain oic.sceneMember resource types.
4493     The additional parameters are
4494     lastScene, this is the scene value last set by any OIC Client
4495     sceneValueList, this is the list of available scenes
4496     lastScene shall be listed in sceneValueList.
4497
4498   get:
4499     description: |
4500       Provides the current list of web links pointing to scenes
4501
4502   responses:
4503     200:
4504       body:
4505         application/json:
4506           schema: |
4507             {
4508               "$schema": "http://json-schema.org/draft-04/schema#",
4509               "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4510               "title": "Scene Collection",
4511               "definitions": {
4512                 "oic.sceneCollection": {
4513                   "type": "object",
4514                   "properties": {
4515                     "lastScene": {
4516                       "type": "string",
4517                       "description": "Last selected Scene, shall be part of
4518 sceneValues",
4519                       "format": "UTF8"
4520

```

```

4521         "sceneValues": {
4522             "type": "string",
4523             "description": "ReadOnly, All available scene
4524 values",
4525             "format": "CSV"
4526         },
4527         "n": {
4528             "type": "string",
4529             "description": "Used to name the Scene collection",
4530             "format": "UTF8"
4531         },
4532         "id": {
4533             "type": "string",
4534             "description": "A unique string that could be a hash or
4535 similarly unique"
4536         },
4537         "rts": {
4538             "type": "string",
4539             "description": "ReadOnly, Defines the list of
4540 allowable resource types in links included in the collection; new links being created can only be
4541 from this list",
4542             "format": "UTF8"
4543         },
4544         "links": {
4545             "type": "array",
4546             "description": "Array of OIC web links that are
4547 reference from this collection",
4548             "items" : {
4549                 "allOf": [
4550                     { "$ref":
4551 "http://openinterconnect.org/schemas/oic.web-link.json#" },
4552                     { "required" : [ "ins" ] }
4553                 ]
4554             }
4555         },
4556         "required": [ "lastScene","sceneValues","rts","id" ]
4557     },
4558     },
4559     "type": "object",
4560     "allOf" : [
4561         { "$ref": "oic.core.json#/definitions/oic.core" },
4562         { "$ref": "#/definitions/oic.sceneCollection" }
4563     ]
4564 }
4565 }
4566 }
4567
4568     example: |
4569     {
4570         "lastScene": "off",
4571         "sceneValues": "off,Reading,TVWatching",
4572         "rt": "oic.wk.sceneCollection",
4573         "n": "My Scenes for my living room",
4574         "id": "0685B960-736F-46F7-BECO-9E6CBD671ADC1",
4575         "rts": "oic.wk.sceneMember",
4576         "links": [
4577             ]
4578     }
4579
4580     post:
4581     description: |
4582     Provides the action to create a new sceneMember in the SceneCollection resource
4583     The only resource type that is allowed to be created is "oic.wk.sceneMember".
4584     The id of the resource will be generated by the implementation.
4585     As example the mappings of the 3 scenes are mapped to different states of an binary switch
4586
4587     body:
4588     application/json

```

```

4589     schema: |
4590         {
4591             "$schema": "http://json-schema.org/draft-04/schema#",
4592             "id": "http://openinterconnect.org/schemas/oic.sceneMember.json#",
4593             "title" : "Scene Member",
4594             "definitions": {
4595                 "oic.sceneMember": {
4596                     "type": "object",
4597                     "properties": {
4598                         "n": {
4599                             "type": "string",
4600                             "description": "Used to name the Scene collection",
4601                             "format": "UTF8"
4602                         },
4603                         "id": {
4604                             "type": "string",
4605                             "description": "Can be an value that is unique to the
4606 use context or a UUIDv4"
4607                         },
4608                         "SceneMappings" : {
4609                             "type": "array",
4610                             "description": "array of mappings per scene, can be 1",
4611                             "items": [
4612                                 {
4613                                     "type": "object",
4614                                     "properties": {
4615                                         "scene": {
4616                                             "type": "string",
4617                                             "description": "Specifies a scene value that will acted upon"
4618                                         },
4619                                         "memberProperty": {
4620                                             "type": "string",
4621                                             "description": "ReadOnly, property name that will be mapped"
4622                                         },
4623                                         "memberValue": {
4624                                             "type": "string",
4625                                             "description": "ReadOnly, value of the Member Property"
4626                                         }
4627                                     },
4628                                     "required": [ "scene", "memberProperty", "memberValue" ]
4629                                 }
4630                             ]
4631                         },
4632                         "link": {
4633                             "type": "string",
4634                             "description": "web link that points at an resource",
4635                             "$ref": "oic.web-link.json#"
4636                         }
4637                     },
4638                     "required": [ "link" ]
4639                 }
4640             },
4641             "type": "object",
4642             "allOf" : [
4643                 { "$ref": "oic.core.json#/definitions/oic.core" },
4644                 { "$ref": "#/definitions/oic.sceneMember" }
4645             ]
4646         }
4647     }
4648
4649
4650     example: |
4651         {
4652             "link": { "href": "coap://mydevice/mybinaryswitch",
4653                     "if": "oic.if.a",
4654                     "rt": "oic.r.switch.binary" },
4655             "n": "my binary switch (for light bulb) mappings",
4656             "sceneMappings": [
4657                 {

```



```

4658         "scene":          "off",
4659         "memberProperty": "value",
4660         "memberValue":    true
4661     },
4662     {
4663         "scene":          "Reading",
4664         "memberProperty": "value",
4665         "memberValue":    false
4666     },
4667     {
4668         "scene":          "TVWatching",
4669         "memberProperty": "value",
4670         "memberValue":    true
4671     }
4672 ]
4673 }
4674
4675 responses:
4676     200:
4677         description: |
4678             Indicates that the target resource was created.
4679             The new resource attributes are provided in the response.
4680
4681         body:
4682             application/json:
4683                 schema: |
4684                     {
4685                         "$schema": "http://json-schema.org/draft-04/schema#",
4686                         "id": "http://openinterconnect.org/schemas/oic.sceneMember.json#",
4687                         "title": "Scene Member",
4688                         "definitions": {
4689                             "oic.sceneMember": {
4690                                 "type": "object",
4691                                 "properties": {
4692                                     "n": {
4693                                         "type": "string",
4694                                         "description": "Used to name the Scene collection",
4695                                         "format": "UTF8"
4696                                     },
4697                                     "id": {
4698                                         "type": "string",
4699                                         "description": "Can be an value that is unique to the
4700 use context or a UUIDv4"
4701                                     },
4702                                     "SceneMappings" : {
4703                                         "type": "array",
4704                                         "description": "array of mappings per scene, can be 1",
4705                                         "items": [
4706                                             {
4707                                                 "type": "object",
4708                                                 "properties": {
4709                                                     "scene": {
4710                                                         "type": "string",
4711                                                         "description": "Specifies a scene value that will acted upon"
4712                                                     },
4713                                                     "memberProperty": {
4714                                                         "type": "string",
4715                                                         "description": "ReadOnly, property name that will be mapped"
4716                                                     },
4717                                                     "memberValue": {
4718                                                         "type": "string",
4719                                                         "description": "ReadOnly, value of the Member Property"
4720                                                     }
4721                                                 },
4722                                                 "required": [ "scene", "memberProperty", "memberValue" ]
4723                                             }
4724                                         ]
4725                                     },

```

```

4726
4727         "link": {
4728             "type": "string",
4729             "description": "web link that points at an resource",
4730             "$ref": "oic.web-link.json#"
4731         }
4732     },
4733     "required": [ "link" ]
4734 }
4735 },
4736
4737 "type": "object",
4738 "allof" : [
4739     { "$ref": "oic.core.json#/definitions/oic.core" },
4740     { "$ref": "#/definitions/oic.sceneMember" }
4741 ]
4742 }
4743

```

```

4744 example: |
4745 {
4746     "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
4747     "n": "my binary switch (for light bulb) mappings",
4748     "link": { "href": "coap://mydevice/mybinaryswitch",
4749             "if": "oic.if.a",
4750             "rt": "oic.r.switch.binary" },
4751     "sceneMappings": [
4752         {
4753             "scene": "off",
4754             "memberProperty": "value",
4755             "memberValue": true
4756         },
4757         {
4758             "scene": "Reading",
4759             "memberProperty": "value",
4760             "memberValue": false
4761         },
4762         {
4763             "scene": "TVWatching",
4764             "memberProperty": "value",
4765             "memberValue": true
4766         }
4767     ]
4768 }
4769

```

```

4770 put:
4771     description: |
4772         Provides the action to change the last settted scene selection.
4773         Calling this method shall update of all sceneMembers to the prescribed membervalue.
4774         When this method is called with the same value as the current lastScene value
4775         then all sceneMembers shall be updated.
4776

```

```

4777 body:
4778     application/json

```

```

4779     schema: |
4780     {
4781         "$schema": "http://json-schema.org/draft-04/schema#",
4782         "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4783         "title" : "Scene Collection",
4784         "definitions": {
4785             "oic.sceneCollection": {
4786                 "type": "object",
4787                 "properties": {
4788                     "lastScene": {
4789                         "type": "string",
4790                         "description": "Last selected Scene, shall be part of
4791 sceneValues",
4792                         "format": "UTF8"

```

```

4793     },
4794     "sceneValues": {
4795         "type": "string",
4796         "description": "ReadOnly, All available scene
values",
4797         "format": "CSV",
4798     },
4799     "n": {
4800         "type": "string",
4801         "description": "Used to name the Scene collection",
4802         "format": "UTF8"
4803     },
4804     "id": {
4805         "type": "string",
4806         "description": "A unique string that could be a hash or
similarly unique"
4807     },
4808     "rts": {
4809         "type": "string",
4810         "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list",
4811         "format": "UTF8"
4812     },
4813     "links": {
4814         "type": "array",
4815         "description": "Array of OIC web links that are
reference from this collection",
4816         "items": {
4817             "allOf": [
4818                 { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
4819                 { "required": [ "ins" ] }
4820             ]
4821         }
4822     },
4823     "required": [ "lastScene" ]
4824 },
4825 "type": "object",
4826 "allOf": [
4827     { "$ref": "oic.core.json#/definitions/oic.core" },
4828     { "$ref": "#/definitions/oic.sceneCollection" }
4829 ]
4830 }
4831
4832 example: |
4833 {
4834     "lastScene": "Reading"
4835 }
4836
4837 responses:
4838     200:
4839         description: |
4840             Indicates that the value is changed.
4841             The changed properties are provided in the response.
4842
4843         body:
4844             application/json:
4845                 schema: |
4846                 {
4847                     "$schema": "http://json-schema.org/draft-04/schema#",
4848                     "id": "http://openinterconnect.org/schemas/oic.sceneCollection.json#",
4849                     "title": "Scene Collection",
4850                     "definitions": {

```

```

4860         "oic.sceneCollection": {
4861             "type": "object",
4862             "properties": {
4863                 "lastScene": {
4864                     "type": "string",
4865                     "description": "Last selected Scene, shall be part of
sceneValues",
4866                     "format": "UTF8"
4867                 },
4868                 "sceneValues": {
4869                     "type": "string",
4870                     "description": "ReadOnly, All available scene
values",
4871                     "format": "CSV"
4872                 },
4873                 "n": {
4874                     "type": "string",
4875                     "description": "Used to name the Scene collection",
4876                     "format": "UTF8"
4877                 },
4878                 "id": {
4879                     "type": "string",
4880                     "description": "A unique string that could be a hash or
similarly unique"
4881                 },
4882                 "rts": {
4883                     "type": "string",
4884                     "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list",
4885                     "format": "UTF8"
4886                 },
4887                 "links": {
4888                     "type": "array",
4889                     "description": "Array of OIC web links that are
reference from this collection",
4890                     "items": {
4891                         "allOf": [
4892                             { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
4893                             { "required" : [ "ins" ] }
4894                         ]
4895                     }
4896                 },
4897                 "required": [ "lastScene" ]
4898             }
4899         },
4900         "type": "object",
4901         "allOf" : [
4902             { "$ref": "oic.core.json#/definitions/oic.core" },
4903             { "$ref": "#/definitions/oic.sceneCollection" }
4904         ]
4905     }
4906
4907     example: |
4908     {
4909         "lastScene": "Reading"
4910     }
4911
4912     delete:
4913     description: |
4914     No change from collection.
4915     When delete is used with the URI of the collection without and query parameters then the
entire collection is deleted
4916     When the delete uses the "ins" parameter with the value of a specific link then only that
link is deleted
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928

```

```

4929     queryParameters:
4930       ins:
4931         type: string
4932         description: Delete the Web link identified by the string - could be a UUID.
4933
4934       required: false
4935       example: DELETE /mycollection?ins="0685B960-FFFF-46F7-BEC0-9E6234671ADC1"
4936
4937     responses:
4938       200:
4939         description: |
4940           The web link instance or the the entire collection has been successfully deleted
4941
4942       400:
4943         description: |
4944           The request is invalid
4945

```

4946 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
sceneValues	string	yes	Read Only	All Available Scene Values
n	string		Read Write	Used to name the Scene collection
id	string	yes	Read Write	A unique string that could be a hash or similarly unique
rts	string	yes	Read Only	Defines The List Of Allowable Resource Types In Links Included In The Collection; New Links Being Created Can Only Be From This List
links	array		Read Write	Array of OIC web links that are reference from this collection
items				

4947 D.12.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/SceneCollectionResURI	put	get	post	delete	

4948 D.13 Scene Member

4949 D.13.1 Introduction

4950 Collection that models a sceneMember.

4951 D.13.2 Wellknown URI

4952 /SceneMemberResURI

4953 D.13.3 Resource Type

4954 The resource type (rt) is defined as: oic.r.switch.binary.

4955 D.13.4 RAML Definition

```

4956 #%RAML 0.8
4957 title: OICScene
4958 version: v1.0-20150630

```

```

4959 traits:
4960   - interface
4961     queryParameters:
4962       if:
4963         enum: ["oic.if.a", "oic.if.ll"]
4964
4965 /SceneMemberResURI:
4966   description: |
4967     Collection that models a sceneMember.
4968
4969   get:
4970     description: |
4971       Provides the scene member
4972
4973   responses:
4974     200:
4975       body:
4976         application/json:
4977           schema: |
4978             {
4979               "$schema": "http://json-schema.org/draft-04/schema#",
4980               "id": "http://openinterconnect.org/schemas/oic.sceneMember.json#",
4981               "title": "Scene Member",
4982               "definitions": {
4983                 "oic.sceneMember": {
4984                   "type": "object",
4985                   "properties": {
4986                     "n": {
4987                       "type": "string",
4988                       "description": "Used to name the Scene collection",
4989                       "format": "UTF8"
4990                     },
4991                     "id": {
4992                       "type": "string",
4993                       "description": "Can be an value that is unique to the
4994 use context or a UUIDv4"
4995                     },
4996                     "SceneMappings" : {
4997                       "type": "array",
4998                       "description": "array of mappings per scene, can be 1",
4999                       "items": [
5000                         {
5001                           "type": "object",
5002                           "properties": {
5003                             "scene": {
5004                               "type": "string",
5005                               "description": "Specifies a scene value that will acted upon"
5006                             },
5007                             "memberProperty": {
5008                               "type": "string",
5009                               "description": "ReadOnly, property name that will be mapped"
5010                             },
5011                             "memberValue": {
5012                               "type": "string",
5013                               "description": "ReadOnly, value of the Member Property"
5014                             }
5015                           },
5016                           "required": [ "scene", "memberProperty", "memberValue" ]
5017                         }
5018                       ]
5019                     },
5020
5021                     "link": {
5022                       "type": "string",
5023                       "description": "web link that points at an resource",

```

```

5024                                     "$ref": "oic.web-link.json#"
5025                                     }
5026                                 },
5027                                 "required": [ "link" ]
5028                             }
5029     },
5030
5031     "type": "object",
5032     "allof" : [
5033         { "$ref": "oic.core.json#/definitions/oic.core" },
5034         { "$ref": "#/definitions/oic.sceneMember" }
5035     ]
5036 }
5037
5038 example: |
5039 {
5040     "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
5041     "n": "my binary switch (for light bulb) mappings",
5042     "link": { "href": "coap://mydevice/mybinaryswitch",
5043              "if": "oic.if.a",
5044              "rt": "oic.r.switch.binary" },
5045     "sceneMappings": [
5046         {
5047             "scene": "off",
5048             "memberProperty": "value",
5049             "memberValue": true
5050         },
5051         {
5052             "scene": "Reading",
5053             "memberProperty": "value",
5054             "memberValue": false
5055         },
5056         {
5057             "scene": "TVWatching",
5058             "memberProperty": "value",
5059             "memberValue": true
5060         }
5061     ]
5062 }
5063

```

5064 D.13.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the Scene collection
id	string		Read Write	Can be an value that is unique to the use context or a UUIDv4
SceneMappings	array		Read Write	array of mappings per scene, can be 1
items				
scene	string	yes	Read Write	Specifies a scene value that will acted upon
memberProperty	string	yes	Read Only	Property Name That Will Be Mapped
memberValue	string	yes	Read Only	Value Of The Member Property
link	string	yes	Read Write	web link that points at an resource

5065 D.13.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/SceneMemberResURI		get			

5066 D.14 Rules (Top level)

5067 D.14.1 Introduction

5068 Toplevel Rule resource. This resource is a generic collection resource The rts value shall contain
5069 oic.wk.rule resource types

5070 D.14.2 Wellknown URI

5071 /RuleListResURI

5072 D.14.3 Resource Type

5073 The resource type (rt) is defined as: oic.wk.ruleList.

5074 D.14.4 RAML Definition

5075 `##RAML 0.8`

5076 `title: OICRules`

5077 `version: v1.0-20150810`

5078 `traits:`

5079 `- interface`

5080 `queryParameters:`

5081 `if:`

5082 `enum: ["oic.if.a", "oic.if.ll"]`

5083

5084 `/RuleListResURI:`

5085 `description: |`

5086 `Toplevel Rule resource.`

5087 `This resource is a generic collection resource`

5088 `The rts value shall contain oic.wk.rule resource types`

5089

5090 `get:`

5091 `description: |`

5092 `Provides the current list of web links pointing to rules`

5093

5094 `responses:`

5095 `200:`

5096 `body:`

5097 `application/json:`

5098 `schema: |`

```
5099     {
5100       "$schema": "http://json-schema.org/draft-04/schema#",
5101       "id": "http://openinterconnect.org/schemas/oic.collection.json#",
5102       "title": "Collection",
5103       "definitions": {
5104         "oic.collection": {
5105           "type": "object",
5106           "properties": {
5107             "n": {
5108               "type": "string",
5109               "description": "Used to name the collection",
5110               "format": "UTF8"
5111             },
5112             "id": {
5113               "oneOf": [
5114                 { "type": "number", "description": "if id
5115 property is an number" },
5116                 { "type": "string", "description": "if id
5117 property is an number" }
5118               ]
5119             },
5120             "rts": {
5121               "type": "string",
```



```

5122                                     "description": "Defines the list of allowable
5123 resource types in links included in the collection; new links being created can only be from this
5124 list",
5125                                     "format": "UTF8"
5126                                     },
5127                                     "links": {
5128                                         "type": "array",
5129                                         "description": "Array of OIC web links that are
5130 reference from this collection",
5131                                         "items" : {
5132                                             "allOf": [
5133                                                 { "$ref":
5134 "http://openinterconnect.org/schemas/oic.web-link.json#" },
5135                                                 { "required" : [ "ins" ] }
5136                                             ]
5137                                         }
5138                                     },
5139                                     "required": [ "links" ]
5140                                 }
5141                             },
5142                             "type": "object",
5143                             "allOf" : [
5144                                 { "$ref": "oic.core.json#/definitions/oic.core" },
5145                                 { "$ref": "#/definitions/oic.collection" }
5146                             ]
5147                         }
5148                     ]
5149                 }
5150
5151     example: |
5152     {
5153         "rt":      "oic.wk.ruleList",
5154         "n":       "list of rules",
5155         "id":      "my_ruleList_1",
5156         "rts":     "oic.wk.rule",
5157         "links": [
5158             ]
5159     }
5160
5161     post:
5162     description: |
5163     Provides the action to create a new rule in the ruleList resource
5164     The only resource type that is allowed to be created "oic.wk.rule".
5165     The example contains a condition, currentStatus and test.
5166
5167     body:
5168     application/json
5169     schema: |
5170     {
5171         "$schema": "http://json-schema.org/draft-04/schema#",
5172         "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5173         "title" : "Rule",
5174         "definitions": {
5175             "oic.rule": {
5176                 "type": "object",
5177                 "properties": {
5178                     "condition": {
5179                         "type": "string",
5180                         "description": "condition of the rule",
5181                         "format": "UTF8"
5182                     },
5183                     "currentStatus": {
5184                         "type": "string",
5185                         "description": "ReadOnly, the current state, can be
5186 one of: enabled, disabled, error"
5187                     }
5188                 },
5189                 "n": {

```

```

5189         "type": "string",
5190         "description": "Used to name the Rule collection",
5191         "format": "UTF8"
5192     },
5193     "test": {
5194         "type": "boolean",
5195         "description": "Indicates initiation of test mode for
the rule"
5196     },
5197     "id": {
5198         "type": "string",
5199         "description": "Can be an value that is unique to the
use context or a UUIDv4"
5200     },
5201     "rts": {
5202         "type": "string",
5203         "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
5204     },
5205     "links": {
5206         "type": "array",
5207         "description": "Array of OIC web links that are the
rule members, this is the script",
5208         "items": {
5209             "allOf": [
5210                 { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
5211                 { "required": [ "ins" ] }
5212             ]
5213         }
5214     },
5215     "required": [ "links", "condition", "currentStatus", "test", "id",
"rts" ]
5216 },
5217 },
5218 "type": "object",
5219 "allOf" : [
5220     { "$ref": "oic.core.json#/definitions/oic.core" },
5221     { "$ref": "#/definitions/oic.rule" }
5222 ]
5223 }
5224
5225 example: |
5226 {
5227     "condition": "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value = true)
and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFFF:tempid:temperature > 30)",
5228     "currentStatus": "off",
5229     "test": false,
5230     "rt": "oic.wk.rule",
5231     "n": "my first rule",
5232     "id": "my_rule1",
5233     "rts": "oic.r.ruleMember",
5234     "links": [
5235     ]
5236 }
5237
5238 responses:
5239 200:
5240 description: |
5241 Indicates that the target resource was created.
5242 The created resource attributes are provided in the response,
5243 including the server generated identifier.
5244
5245 body:
5246 application/json:

```

```

5257     schema: |
5258         {
5259             "$schema": "http://json-schema.org/draft-04/schema#",
5260             "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5261             "title": "Rule",
5262             "definitions": {
5263                 "oic.rule": {
5264                     "type": "object",
5265                     "properties": {
5266                         "condition": {
5267                             "type": "string",
5268                             "description": "condition of the rule",
5269                             "format": "UTF8"
5270                         },
5271                         "currentStatus": {
5272                             "type": "string",
5273                             "description": "ReadOnly, the current state, can be
5274 one of: enabled, disabled, error"
5275                         },
5276                         "n": {
5277                             "type": "string",
5278                             "description": "Used to name the Rule collection",
5279                             "format": "UTF8"
5280                         },
5281                         "test": {
5282                             "type": "boolean",
5283                             "description": "Indicates initiation of test mode for
5284 the rule"
5285                         },
5286                         "id": {
5287                             "type": "string",
5288                             "description": "Can be an value that is unique to the
5289 use context or a UUIDv4"
5290                         },
5291                         "rts": {
5292                             "type": "string",
5293                             "description": "ReadOnly, Defines the list of
5294 allowable resource types in links included in the collection; new links being created can only be
5295 from this list"
5296                         },
5297                         "links": {
5298                             "type": "array",
5299                             "description": "Array of OIC web links that are the
5300 rule members, this is the script",
5301                             "items": {
5302                                 "allOf": [
5303                                     { "$ref":
5304 "http://openinterconnect.org/schemas/oic.web-link.json#" },
5305                                     { "required": [ "ins" ] }
5306                                 ]
5307                             }
5308                         },
5309                         "required": [ "links", "condition", "currentStatus", "test", "id",
5310 "rts" ]
5311                     }
5312                 },
5313                 "type": "object",
5314                 "allOf": [
5315                     { "$ref": "oic.core.json#/definitions/oic.core" },
5316                     { "$ref": "#/definitions/oic.rule" }
5317                 ]
5318             }
5319         }
5320     }
5321
5322     example: |
5323         {
5324             "condition": "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value == true)
5325 and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFF:tempid:temperature > 30)",

```

```

5326         "currentStatus": "off",
5327         "test":         false,
5328         "rt":           "oic.wk.rule",
5329         "n":            "my first rule",
5330         "id":           "FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1" ,
5331         "rts":          "oic.r.ruleMember",
5332         "links": [
5333             ]
5334     }
5335
5336     delete:
5337         description: |
5338             No change from collection.
5339             When delete is used with the URI of the collection without and query parameters then the
5340             entire collection is deleted
5341             When the delete uses the "ins" parameter with the value of a specific link then only that
5342             link is deleted
5343
5344         queryParameters:
5345             ins:
5346                 type: string
5347                 description: Delete the Web link identified by the string - could be a UUID.
5348
5349             required: false
5350             example: DELETE /mycollection?ins="FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1"
5351
5352         responses:
5353             200:
5354                 description: |
5355                     The web link instance or the the entire collection has been successfully deleted
5356
5357             400:
5358                 description: |
5359                     The request is invalid
5360

```

5361 D.14.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the collection
id	object			
rts	string		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
links	array	yes	Read Write	Array of OIC web links that are reference from this collection
items				

5362 D.14.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/RuleListResURI		get	post	delete	

5363 D.15 Rule

5364 D.15.1 Introduction

5365 Collection that models a rule. This resource is an generic collection resource with additional
5366 parameters. The rts value shall contain oic.wk.ruleMember resource types. The additional

5367 parameters are condition, this is the rule that will be evaluated currentStatus, the current state of
5368 the rule, can be "enabled, disabled, error" test an trigger once activation of the rule

5369 **D.15.2 Wellknown URI**

5370 /RuleResURI

5371 **D.15.3 Resource Type**

5372 The resource type (rt) is defined as: oic.wk.rule.

5373 **D.15.4 RAML Definition**

```
5374 #%RAML 0.8
5375 title: OICRules
5376 version: v1.0-20150810
5377 traits:
5378   - interface
5379     queryParameters:
5380       if:
5381         enum: ["oic.if.a", "oic.if.ll"]
5382
5383 /RuleResURI:
5384   description: |
5385     Collection that models a rule.
5386     This resource is an generic collection resource with additional parameters.
5387     The rts value shall contain oic.wk.ruleMember resource types.
5388     The additional parameters are
5389     condition, this is the rule that will be evaluated
5390     currentStatus, the current state of the rule, can be "enabled, disabled, error"
5391     test an trigger once activation of the rule
5392
5393   get:
5394     description: |
5395       Provides the current rule and list of web links to the rule members
5396
5397   responses:
5398     200:
5399       body:
5400         application/json:
5401           schema: |
5402             {
5403               "$schema": "http://json-schema.org/draft-04/schema#",
5404               "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5405               "title": "Rule",
5406               "definitions": {
5407                 "oic.rule": {
5408                   "type": "object",
5409                   "properties": {
5410                     "condition": {
5411                       "type": "string",
5412                       "description": "condition of the rule",
5413                       "format": "UTF8"
5414                     },
5415                     "currentStatus": {
5416                       "type": "string",
5417                       "description": "ReadOnly, the current state, can be
5418 one of: enabled, disabled, error"
5419                     },
5420                     "n": {
5421                       "type": "string",
5422                       "description": "Used to name the Rule collection",
5423                       "format": "UTF8"
5424                     }
5425                   }
5426                 }
5427             }
```

```

5425         "test": {
5426             "type": "boolean",
5427             "description": "Indicates initiation of test mode for
the rule"
5428         },
5429     },
5430     "id": {
5431         "type": "string",
5432         "description": "Can be an value that is unique to the
use context or a UUIDv4"
5433     },
5434     "rts": {
5435         "type": "string",
5436         "description": "ReadOnly, Defines the list of
allowable resource types in links included in the collection; new links being created can only be
from this list"
5437     },
5438     "links": {
5439         "type": "array",
5440         "description": "Array of OIC web links that are the
rule members, this is the script",
5441         "items": {
5442             "allOf": [
5443                 { "$ref":
"http://openinterconnect.org/schemas/oic.web-link.json#" },
5444                 { "required": [ "ins" ] }
5445             ]
5446         }
5447     },
5448     "required": [ "links", "condition", "currentStatus", "test", "id",
"rts" ]
5449 },
5450 },
5451 },
5452 },
5453 },
5454 },
5455 },
5456 },
5457 },
5458 },
5459 },
5460 },
5461 },
5462 },
5463 },
5464 },
5465 },
5466 example: |
5467 {
5468     "condition": "(FFFAB960-736F-46F7-BEC0-9E6CBD671ADC1:binaryswitchid:value == true)
and (FFFFB960-736F-46F7-BEC0-9E6CBD671FFFF:tempid:temperature > 30)",
5469     "currentStatus": "disabled",
5470     "test": false,
5471     "rt": "oic.wk.rule",
5472     "n": "my first rule",
5473     "id": "FFFFB960-736F-46F7-BEC0-9E6CBD671ADC1",
5474     "rts": "oic.r.ruleMember",
5475     "links": [
5476     ]
5477 }
5478 }
5479 }
5480 post:
5481 description: |
5482 Provides the action to create a new ruleMember in the rule resource
5483 The only resource type that is allowed to be created is "oic.wk.ruleMember".
5484 The id of the resource will be generated by the implementation.
5485
5486 body:
5487 application/json
5488 schema: |
5489 {
5490     "$schema": "http://json-schema.org/draft-04/schema#",
5491     "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",

```

```

5492     "title" : "Rule Member",
5493     "definitions": {
5494         "oic.ruleMember": {
5495             "type": "object",
5496             "properties": {
5497                 "n": {
5498                     "type": "string",
5499                     "description": "Used to name the Rule member",
5500                     "format": "UTF8"
5501                 },
5502                 "id": {
5503                     "type": "string",
5504                     "description": "Can be an value that is unique to the use context or a UUIDv4"
5505                 },
5506                 "memberProperty": {
5507                     "type": "string",
5508                     "description": "ReadOnly, property name that will be mapped"
5509                 },
5510                 "memberValue": {
5511                     "oneOf" : [
5512                         { "type": "number", "description": "if
5513 member property is an number" },
5514                         { "type": "string", "description": "if
5515 member property is an number" },
5516                         { "type": "boolean", "description": "if
5517 member property is an boolean" }
5518                     ],
5519                     "description": "ReadOnly, value of the Member Property"
5520                 },
5521                 "link": {
5522                     "type": "string",
5523                     "description": "web link that points at a resource",
5524                     "$ref": "oic.web-link.json#"
5525                 }
5526             },
5527             "required": [ "id", "link", "memberProperty", "memberValue" ],
5528             "additionalProperties": false
5529         }
5530     },
5531     "type": "object",
5532     "allOf" : [
5533         { "$ref": "oic.core.json#/definitions/oic.core" },
5534         { "$ref": "#/definitions/oic.ruleMember" }
5535     ]
5536 }
5537
5538
5539     example: |
5540     {
5541         "link": { "href": "coap://mydevice/mybinaryswitch",
5542                 "if": "oic.if.a",
5543                 "rt": "oic.r.switch.binary" },
5544         "id": "",
5545         "n": "my binary switch (for light bulb) mappings",
5546         "memberProperty": "value",
5547         "memberValue": true
5548     }
5549
5550     responses:
5551     200:
5552         description: |
5553             Indicates that the target resource was created.
5554             The new resource attributes are provided in the response.
5555
5556         body:
5557         application/json:
5558             schema: |

```

```

5559     {
5560         "$schema": "http://json-schema.org/draft-04/schema#",
5561         "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",
5562         "title": "Rule Member",
5563         "definitions": {
5564             "oic.ruleMember": {
5565                 "type": "object",
5566                 "properties": {
5567                     "n": {
5568                         "type": "string",
5569                         "description": "Used to name the Rule member",
5570                         "format": "UTF8"
5571                     },
5572                     "id": {
5573                         "type": "string",
5574                         "description": "Can be an value that is unique to the use context or a
5575 UUIDv4"
5576                     },
5577                     "memberProperty": {
5578                         "type": "string",
5579                         "description": "ReadOnly, property name that will be mapped"
5580                     },
5581                     "memberValue": {
5582                         "oneOf": [
5583                             { "type": "number", "description": "if
5584 member property is an number" },
5585                             { "type": "string", "description": "if
5586 member property is an number" },
5587                             { "type": "boolean", "description": "if
5588 member property is an boolean" }
5589                         ],
5590                         "description": "ReadOnly, value of the Member Property"
5591                     },
5592                     "link": {
5593                         "type": "string",
5594                         "description": "web link that points at a resource",
5595                         "$ref": "oic.web-link.json#"
5596                     }
5597                 },
5598                 "required": [ "id", "link", "memberProperty", "memberValue" ],
5599                 "additionalProperties": false
5600             }
5601         },
5602         "type": "object",
5603         "allOf": [
5604             { "$ref": "oic.core.json#/definitions/oic.core" },
5605             { "$ref": "#/definitions/oic.ruleMember" }
5606         ]
5607     }
5608 }
5609
5610 example: |
5611 {
5612     "id": "FFFFB960-FFFF-46F7-BEC0-9E6234671ADC1",
5613     "n": "my binary switch (for light bulb) mappings",
5614     "link": { "href": "coap://mydevice/mybinaryswitch",
5615         "if": "oic.if.a",
5616         "rt": "oic.r.switch.binary" },
5617     "memberProperty": "value",
5618     "memberValue": true
5619 }
5620
5621 put:
5622     description: |
5623     Provides the action to enable/disable the rule and an test mode for the rule.
5624     Calling this method with test = true will update of all ruleMembers to the prescribed
5625     membervalue.
5626

```



```

5627     body:
5628     application/json
5629         schema: |
5630             {
5631                 "$schema": "http://json-schema.org/draft-04/schema#",
5632                 "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5633                 "title": "Rule",
5634                 "definitions": {
5635                     "oic.rule": {
5636                         "type": "object",
5637                         "properties": {
5638                             "condition": {
5639                                 "type": "string",
5640                                 "description": "condition of the rule",
5641                                 "format": "UTF8"
5642                             },
5643                             "currentStatus": {
5644                                 "type": "string",
5645                                 "description": "ReadOnly, the current state, can be
5646 one of: enabled, disabled, error"
5647                             },
5648                             "n": {
5649                                 "type": "string",
5650                                 "description": "Used to name the Rule collection",
5651                                 "format": "UTF8"
5652                             },
5653                             "test": {
5654                                 "type": "boolean",
5655                                 "description": "Indicates initiation of test mode for
5656 the rule"
5657                             },
5658                             "id": {
5659                                 "type": "string",
5660                                 "description": "Can be an value that is unique to the
5661 use context or a UUIDv4"
5662                             },
5663                             "rts": {
5664                                 "type": "string",
5665                                 "description": "ReadOnly, Defines the list of
5666 allowable resource types in links included in the collection; new links being created can only be
5667 from this list"
5668                             },
5669                             "links": {
5670                                 "type": "array",
5671                                 "description": "Array of OIC web links that are the
5672 rule members, this is the script",
5673                                 "items": {
5674                                     "allOf": [
5675                                         { "$ref": "http://openinterconnect.org/schemas/oic.web-link.json#" },
5676                                         { "required": [ "ins" ] }
5677                                     ]
5678                                 }
5679                             },
5680                             "required": [ "currentStatus" ]
5681                         },
5682                     }
5683                 },
5684             },
5685             "type": "object",
5686             "allOf": [
5687                 { "$ref": "oic.core.json#/definitions/oic.core" },
5688                 { "$ref": "#/definitions/oic.rule" }
5689             ]
5690         }
5691     example: |
5692         {
5693             "currentStatus": "enabled"
5694         }
5695

```

```

5696     }
5697
5698     responses:
5699         200:
5700             description: |
5701                 Indicates that the value is changed.
5702                 The changed properties are provided in the response.
5703
5704             body:
5705                 application/json:
5706                     schema: |
5707                         {
5708                             "$schema": "http://json-schema.org/draft-04/schema#",
5709                             "id": "http://openinterconnect.org/schemas/oic.rule.json#",
5710                             "title" : "Rule",
5711                             "definitions": {
5712                                 "oic.rule": {
5713                                     "type": "object",
5714                                     "properties": {
5715                                         "condition": {
5716                                             "type": "string",
5717                                             "description": "condition of the rule",
5718                                             "format": "UTF8"
5719                                         },
5720                                         "currentStatus": {
5721                                             "type": "string",
5722                                             "description": "ReadOnly, the current state, can be
5723 one of: enabled, disabled, error"
5724                                         },
5725                                         "n": {
5726                                             "type": "string",
5727                                             "description": "Used to name the Rule collection",
5728                                             "format": "UTF8"
5729                                         },
5730                                         "test": {
5731                                             "type": "boolean",
5732                                             "description": "Inidcates initiation of test mode for
5733 the rule"
5734                                         },
5735                                         "id": {
5736                                             "type": "string",
5737                                             "description": "Can be an value that is unique to the
5738 use context or a UUIDv4"
5739                                         },
5740                                         "rts": {
5741                                             "type": "string",
5742                                             "description": "ReadOnly, Defines the list of
5743 allowable resource types in links included in the collection; new links being created can only be
5744 from this list"
5745                                         },
5746                                         "links": {
5747                                             "type": "array",
5748                                             "description": "Array of OIC web links that are the
5749 rule members, this is the script",
5750                                             "items" : {
5751                                                 "allOf": [
5752                                                     { "$ref":
5753 "http://openinterconnect.org/schemas/oic.web-link.json#" },
5754                                                     { "required" : [ "ins" ] }
5755                                                 ]
5756                                             }
5757                                         },
5758                                         "required": [ "currentStatus" ]
5759                                     }
5760                                 }
5761                             },
5762                             "type": "object",

```

```

5764         "allof" : [
5765             { "$ref": "oic.core.json#/definitions/oic.core" },
5766             { "$ref": "#/definitions/oic.rule" }
5767         ]
5768     }
5769
5770     example: |
5771         {
5772             "currentStatus": "enabled"
5773         }
5774
5775     delete:
5776         description: |
5777             No change from collection.
5778             When delete is used with the URI of the collection without and query parameters then the
5779             entire collection is deleted
5780             When the delete uses the "ins" parameter with the value of a specific link then only that
5781             link is deleted
5782
5783     queryParameters:
5784         ins:
5785             type: string
5786             description: Delete the Web link identified by the string - could be a UUID.
5787
5788         required: false
5789         example: DELETE /mycollection?ins="FFFFB960-FFFF-46F7-BEC0-9E6234671ADC1"
5790
5791     responses:
5792         200:
5793             description: |
5794                 The web link instance or the the entire collection has been successfully deleted
5795
5796         400:
5797             description: |
5798                 The request is invalid
5799
5800

```

D.15.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
condition	string	yes	Read Write	condition of the rule
currentStatus	string	yes	Read Only	The Current State, Can Be One Of: Enabled, Disabled, Error
n	string		Read Write	Used to name the Rule collection
test	boolean	yes	Read Write	Indicates initiation of test mode for the rule
id	string	yes	Read Write	Can be an value that is unique to the use context or a UUIDv4
rts	string	yes	Read Only	Defines The List Of Allowable Resource Types In Links Included In The Collection; New Links Being Created Can Only Be From This List
links	array	yes	Read Write	Array of OIC web links that are the rule members, this is the script
items				

5801 **D.15.6 CRUDN behavior**

Resource	Create	Read	Update	Delete	Notify
/RuleResURI	put	get	post	delete	

5802 **D.16 Rule Member**

5803 **D.16.1 Introduction**

5804 Rule member resource. This resource is assignment statement of an property in a resource
5805 indicated by an URI

5806 **D.16.2 Wellknown URI**

5807 /RuleMemberResURI

5808 **D.16.3 Resource Type**

5809 The resource type (rt) is defined as: oic.r.switch.binary.

5810 **D.16.4 RAML Definition**

```
5811 #%RAML 0.8
5812 title: OICRules
5813 version: v1.0-20150810
5814 traits:
5815   - interface
5816     queryParameters:
5817       if:
5818         enum: ["oic.if.a", "oic.if.ll"]
5819
5820 /RuleMemberResURI:
5821   description: |
5822     Rule member resource.
5823     This resource is assignment statement of an property in a resource indicated by an URI
5824
5825   get:
5826     description: |
5827       Provides the rule mappings
5828
5829   responses:
5830     200:
5831       body:
5832         application/json:
5833           schema: |
5834             {
5835               "$schema": "http://json-schema.org/draft-04/schema#",
5836               "id": "http://openinterconnect.org/schemas/oic.ruleMember.json#",
5837               "title": "Rule Member",
5838               "definitions": {
5839                 "oic.ruleMember": {
5840                   "type": "object",
5841                   "properties": {
5842                     "n": {
5843                       "type": "string",
5844                       "description": "Used to name the Rule member",
5845                       "format": "UTF8"
5846                     },
5847                     "id": {
5848                       "type": "string",
5849                       "description": "Can be an value that is unique to the use context or a
5850 UUIDv4"
5851                     }
5852                   }
5853             }
```

```

5852         "memberProperty": {
5853             "type": "string",
5854             "description": "ReadOnly, property name that will be mapped"
5855         },
5856         "memberValue": {
5857             "oneOf" : [
5858                 { "type": "number", "description": "if
5859 member property is an number" },
5860                 { "type": "string", "description": "if
5861 member property is an number" },
5862                 { "type": "boolean", "description": "if
5863 member property is an boolean" }
5864             ],
5865             "description": "ReadOnly, value of the Member Property"
5866         },
5867         "link": {
5868             "type": "string",
5869             "description": "web link that points at a resource",
5870             "$ref": "oic.web-link.json#"
5871         }
5872     },
5873     "required": [ "id", "link", "memberProperty", "memberValue" ],
5874     "additionalProperties": false
5875 }
5876 },
5877
5878 "type": "object",
5879 "allOf" : [
5880     { "$ref": "oic.core.json#/definitions/oic.core" },
5881     { "$ref": "#/definitions/oic.ruleMember" }
5882 ]
5883 }
5884

```

```

5885 example: |
5886 {
5887     "id": "FFFFB960-FFFF-46F7-BEC0-9E6234671ADC1",
5888     "n": "my binary switch (for light bulb) mappings",
5889     "link": { "href": "coap://mydevice/mybinaryswitch",
5890             "if": "oic.if.a",
5891             "rt": "oic.r.switch.binary" },
5892     "memberProperty": "value",
5893     "memberValue": true
5894 }
5895

```

5896 D.16.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
n	string		Read Write	Used to name the Rule member
id	string	yes	Read Write	Can be an value that is unique to the use context or a UUIDv4
memberProperty	string	yes	Read Only	Property Name That Will Be Mapped
memberValue	object	yes		
link	string	yes	Read Write	web link that points at a resource

5897 D.16.6 CRUDN behavior

Resource	Create	Read	Update	Delete	Notify
/RuleMemberResURI		get			

5898