



OIC CORE SPECIFICATION V1.1.1 Part 1

Open Connectivity Foundation (OCF)
admin@openconnectivity.org

Legal Disclaimer

4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

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

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

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

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

CONTENTS

22
23
24
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

1	Scope	11
2	Normative references	11
3	Terms, definitions, symbols and abbreviations	14
3.1	Terms and definitions	14
3.2	Symbols and abbreviations	16
3.3	Conventions	17
3.4	Data types	17
4	Document conventions and organization	18
5	Architecture.....	19
5.1	Overview	19
5.2	Principle	19
5.3	Functional block diagram.....	21
5.3.1	Framework	22
5.4	Example Scenario with roles.....	22
5.5	Example Scenario: Bridging to Non- OCF ecosystem.....	23
6	Identification and addressing.....	24
6.1	Introduction	24
6.2	Identification.....	25
6.2.1	Resource identification and addressing	25
6.3	Namespace:	26
6.4	Network addressing	26
7	Resource model	26
7.1	Introduction	26
7.2	Resource.....	27
7.3	Property	28
7.3.1	Introduction	28
7.3.2	Common Properties.....	29
7.4	Resource Type	30
7.4.1	Introduction	30
7.4.2	Resource Type Property.....	31
7.4.3	Resource Type definition.....	31
7.5	Device Type	32
7.6	Interface	33
7.6.1	Introduction	33
7.6.2	Interface Property	33
7.6.3	Interface methods	34
7.7	Resource representation	42
7.8	Structure	42
7.8.1	Introduction	42
7.8.2	Resource Relationships.....	42
7.8.3	Collections	48

65	8	CRUDN	51
66	8.1	Overview	51
67	8.2	CREATE	52
68	8.2.1	CREATE request	53
69	8.2.2	Processing by the Server	53
70	8.2.3	CREATE response	53
71	8.3	RETRIEVE	53
72	8.3.1	RETRIEVE request.....	54
73	8.3.2	Processing by the Server	54
74	8.3.3	RETRIEVE response	54
75	8.4	UPDATE	54
76	8.4.1	UPDATE request	55
77	8.4.2	Processing by the Server	55
78	8.4.3	UPDATE response	55
79	8.5	DELETE	55
80	8.5.1	DELETE request	56
81	8.5.2	Processing by the Server	56
82	8.5.3	DELETE response.....	56
83	8.6	NOTIFY	56
84	9	Network and connectivity	56
85	9.1	Introduction	56
86	9.2	Architecture	57
87	9.3	• A node may translate and route messaging between IPv6 and non-IPv6 networks.IPv6 network layer requirements.....	58
88			
89	9.3.1	Introduction	58
90	9.3.2	IPv6 node requirements	58
91	9.3.3	IPv6 constrained nodes	59
92	10	Endpoint discovery.....	60
93	10.1	Introduction	60
94	10.2	CoAP based Endpoint discovery	60
95	11	Functional interactions	61
96	11.1	Introduction	61
97	11.2	Provisioning.....	61
98	11.3	Resource discovery	65
99	11.3.1	Introduction	65
100	11.3.2	Resource based discovery: mechanisms	65
101	11.3.3	Resource based discovery: Information publication process	67
102	11.3.4	Resource based discovery: Finding information.....	68
103	11.3.5	Resource discovery using /oic/res	73
104	11.3.6	Resource directory (RD) based discovery.....	74
105	11.4	Notification	81
106	11.4.1	Overview	81
107	11.4.2	Observe	81
108	11.5	Device management	83

109	11.5.1	Diagnostics and maintenance	83
110	11.6	Scenes	84
111	11.6.1	Introduction	84
112	11.6.2	Scenes	85
113	11.6.3	Security considerations	88
114	12	Messaging.....	89
115	12.1	Introduction	89
116	12.2	Mapping of CRUDN to CoAP	89
117	12.2.1	Overview.....	89
118	12.2.2	URIs.....	89
119	12.2.3	CoAP method with request and response	89
120	12.2.4	Content Type negotiation	91
121	12.2.5	CRUDN to CoAP response codes	92
122	12.2.6	CoAP block transfer	92
123	12.2.7	CoAP serialization over TCP	92
124	12.3	Payload Encoding in CBOR	94
125	13	Security.....	94
126	14	Multi resource model support	94
127	14.1	Interoperability issue	94
128	14.1.1	Multiple IoT Standards	94
129	14.1.2	Different resource models	95
130	14.2	A scheme to exchange resource model information	97
131	14.2.1	A scheme to exchange resource model information	97
132	Annex A (informative)	Operation Examples.....	98
133	A.1	Introduction	98
134	A.2	When at home: From smartphone turn on a single light	98
135	A.3	GroupAction execution	99
136	A.4	When garage door opens, turn on lights in hall; also notify smartphone	99
137	A.5	Device management	99
138	Annex B (informative)	OCF interaction scenarios and deployment models	101
139	B.1	OCF interaction scenarios	101
140	B.2	Deployment model.....	102
141	Annex C (informative)	Other Resource Models and OCF Mapping	104
142	C.1	Multiple resource models	104
143	C.2	OCF approach for support of multiple resource models.....	104
144	C.3	Resource model indication.....	105
145	C.4	An Example Profile (IPSO profile).....	105
146	C.4.1	Conceptual equivalence	105
147	Annex D (normative)	Resource Type definitions	108
148	D.1	List of resource type definitions	108
149	D.2	OCF Collection	108
150	D.2.1	Introduction	108
151	D.2.2	Fixed URI	108

152	D.2.3	Resource Type	108
153	D.2.4	RAML Definition	109
154	D.2.5	Property Definition	116
155	D.2.6	CRUDN Behaviour	118
156	D.2.7	Referenced JSON schemas.....	118
157	D.2.8	oic.oic-link-schema.json	118
158	D.3	OIC Configuration.....	120
159	D.3.1	Introduction	120
160	D.3.2	Fixed URI.....	120
161	D.3.3	Resource Type	120
162	D.3.4	RAML Definition	120
163	D.3.5	Property Definition	123
164	D.3.6	CRUDN Behaviour	123
165	D.4	Device.....	124
166	D.4.1	Introduction	124
167	D.4.2	Fixed URI.....	124
168	D.4.3	Resource Type	124
169	D.4.4	RAML Definition	124
170	D.4.5	Property Definition	125
171	D.4.6	CRUDN Behaviour	125
172	D.5	Maintenance.....	125
173	D.5.1	Introduction	125
174	D.5.2	Fixed URI.....	125
175	D.5.3	Resource Type	126
176	D.5.4	RAML Definition	126
177	D.5.5	Property Definition	128
178	D.5.6	CRUDN Behaviour	128
179	D.6	Platform.....	129
180	D.6.1	Introduction	129
181	D.6.2	Fixed URI.....	129
182	D.6.3	Resource Type	129
183	D.6.4	RAML Definition	129
184	D.6.5	Property Definition	131
185	D.6.6	CRUDN Behaviour	131
186	D.7	Ping.....	132
187	D.7.1	Introduction	132
188	D.7.2	Fixed URI.....	132
189	D.7.3	Resource Type	132
190	D.7.4	RAML Definition	132
191	D.7.5	Property Definition	133
192	D.7.6	CRUDN Behaviour	133
193	D.8	Discoverable Resources.....	133
194	D.8.1	Introduction	133
195	D.8.2	Fixed URI.....	133

196	D.8.3	Resource Type	133
197	D.8.4	RAML Definition	133
198	D.8.5	Property Definition	135
199	D.8.6	CRUDN Behaviour	135
200	D.9	Scenes (Top level)	135
201	D.9.1	Introduction	135
202	D.9.2	Fixed URI	135
203	D.9.3	Resource Type	135
204	D.9.4	RAML Definition	135
205	D.9.5	Property Definition	138
206	D.9.6	CRUDN Behaviour	139
207	D.10	Scene Collections	139
208	D.10.1	Introduction	139
209	D.10.2	Fixed URI	139
210	D.10.3	Resource Type	139
211	D.10.4	RAML Definition	139
212	D.10.5	Property Definition	143
213	D.10.6	CRUDN Behaviour	144
214	D.11	Scene Member	144
215	D.11.1	Introduction	144
216	D.11.2	Fixed URI	144
217	D.11.3	Resource Type	144
218	D.11.4	RAML Definition	144
219	D.11.5	Property Definition	146
220	D.11.6	CRUDN Behaviour	146
221	D.12	Resource directory resource	146
222	D.12.1	Introduction	146
223	D.12.2	Fixed URI	146
224	D.12.3	Resource Type	146
225	D.12.4	RAML Definition	146
226	D.12.5	Property Definition	151
227	D.12.6	CRUDN Behaviour	151
228			
229			

230
231
232

Figures

233	Figure 1: Architecture - concepts	20
234	Figure 2: Functional block diagram	21
235	Figure 3: Communication layering model	22
236	Figure 4: Example illustrating the Roles.....	23
237	Figure 5: Framework - Architecture Detail.....	23
238	Figure 6: Server bridging to Non- OCF device	24
239	Figure 7: Example of a Resource.....	28
240	Figure 8: Example - "Heater" Resource (for illustration only)	40
241	Figure 9: Example - Actuator Interface	40
242	Figure 10: Example of a Link	42
243	Figure 11: Example of distinct Links	42
244	Figure 12: Example of use of anchor in Link	43
245	Figure 13: Example "list of Links"	47
246	Figure 14: List of Links in a Resource.....	47
247	Figure 15: Example showing parts of Collection and Links.....	49
248	Figure 16: Example Collection with simple links (JSON)	49
249	Figure 17: Example Collection with tagged Links (JSON).....	50
250	Figure 18. CREATE operation	53
251	Figure 19. RETRIEVE operation	54
252	Figure 20. UPDATE operation	54
253	Figure 21. DELETE operation	55
254	Figure 22. High Level Network & Connectivity Architecture.....	57
255	Figure 23. Provisioning State Changes.....	62
256	Figure 24. Interactions initiated by the Device to retrieve its configuration from a	
257	configuration source	63
258	Figure 25. Interactions for retrieving the configuration state of an Device.	64
259	Figure 26. Update of and Device configuration	64
260	Figure 27. Resource based discovery: Information publication process.....	68
261	Figure 28. Resource based discovery: Finding information	68
262	Figure 29. Indirect discovery of resource by resource directory	75
263	Figure 30. RD discovery and RD supported query of resources support.....	76
264	Figure 31. Resource Direction Deployment Scenarios	77
265	Figure 32. Observe Mechanism	82
266	Figure 33 Generic scene resource structure	85
267	Figure 34 Interactions to check Scene support and setup of specific scenes	86
268	Figure 35 Client interactions on a specific scene	87
269	Figure 36 Interaction overview due to a Scene change	88

270	Figure 37. When at home: from smartphone turn on a single light.....	99
271	Figure 38. Device management (maintenance)	100
272	Figure 39. Direct interaction between Server and Client	101
273	Figure 40. Interaction between Client and Server using another Server	101
274	Figure 41. Interaction between Client and Server using Intermediary.....	101
275	Figure 42. Interaction between Client and Server using support from multiple Servers and	
276	Intermediary	102
277	Figure 43. Example of Devices	102
278		
279		
280		

Tables

281		
282		
283	Table 1. Data type definition	17
284	Table 2. Name Property Definition	30
285	Table 3. Resource Identity Property Definition	30
286	Table 4. Resource Type Common Property definition	31
287	Table 5. Example foobar Resource Type	32
288	Table 6. Example foobar properties	32
289	Table 7. Resource Interface Property definition	33
290	Table 8. OCF standard Interfaces	34
291	Table 9: Common Properties for Collections (in addition to Common Properties defined in	
292	section 7.3.2)	51
293	Table 10. Parameters of CRUDN messages	52
294	Table 11. List of Core Resources	61
295	Table 12. Configuration Resources	65
296	Table 13. oic.wk.con resource type definition	65
297	Table 14. Mandatory discovery Core Resources	69
298	Table 15. oic.wk.res resource type definition	70
299	Table 16. Protocol scheme registry	70
300	Table 17. oic.wk.d resource type definition	71
301	Table 18. oic.wk.p resource type definition	72
302	Table 19: Selection parameters	78
303	Table 20. Optional diagnostics and maintenance device management Core Resources	83
304	Table 21. oic.wk.mnt resource type definition	83
305	Table 22 list of resource types for Scenes	88
306	Table 23. CoAP request and response	89
307	Table 24. Content Types and Content Formats	91
308	Table 25. Ping resource	93
309	Table 26. oic.wk.ping resource type definition	93
310	Table 27. oic.example.light resource type definition	98
311	Table 28. oic.example.garagedoor resource type definition	98
312	Table 29. Light control resource type definition	106
313	Table 30. Light control resource type definition	106
314	Table 31. Alphabetized list of core resources	108
315		
316		

317 1 Scope

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

- 319 • Core Specification documents: The Core Specification documents specify the Framework, i.e.,
320 the OCF core architecture, interfaces, protocols and services to enable OCF profiles
321 implementation for Internet of Things (IoT) usages and ecosystems.
- 322 • Vertical Profiles Specification documents: The Vertical Profiles Specification documents
323 specify the OCF profiles to enable IoT usages for different market segments such as smart
324 home, industrial, healthcare, and automotive. The Application Profiles Specification is built
325 upon the interfaces and network security of the OCF core architecture defined in the Core
326 Specification.

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

328

329 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

410

411 **3 Terms, definitions, symbols and abbreviations**

412 **3.1 Terms and definitions**

413 **3.1.1**

414 **Client**

415 a logical entity that accesses a Resource on a Server

416 **3.1.2**

417 **Collection**

418 a Resource that contains zero or more Links

419 **3.1.3**

420 **Configuration Source**

421 a Cloud or Service Network or a local read-only file which contains and provides configuration
422 related information to the Devices

423 **3.1.4**

424 **Core Resources**

425 those Resources that are defined in this specification

426 **3.1.5**

427 **Default Interface**

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

429 **3.1.6**

430 **Device**

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

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

433 **3.1.7**

434 **Device Type**

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

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

438 **3.1.8**

439 **Entity**

440 an element of the physical world that is exposed through a Device

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

442 **3.1.9**

443 **Framework**

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

446 **3.1.10**

447 **Links**

448 extends typed web links as specified in IETF RFC 5988

449 **3.1.11**

450 **Non-OCF Device**

451 A device which does not comply with the OCF Device requirements

452 **3.1.12**

453 **Notification**

454 the mechanism to make a Client aware of resource state changes in a Resource

455 **3.1.13**
456 **Observe**
457 the act of monitoring a Resource by sending a RETRIEVE request which is cached by the Server
458 hosting the Resource and reprocessed on every change to that Resource

459 **3.1.14**
460 **Parameter**
461 an element that provides metadata about a Resource referenced by the target URI of a Link

462 **3.1.15**
463 **Partial UPDATE**
464 an UPDATE request to a Resource that includes a subset of the Properties that are visible via the
465 Interface being applied for the Resource Type

466 **3.1.16**
467 **Platform**
468 a physical device containing one or more Devices

469 **3.1.17**
470 **Remote Access Endpoint (RAE) Client**
471 a Client which supports XMPP functionality in order to access a Server from a remote location

472 **3.1.18**
473 **Remote Access Endpoint (RAE) Server**
474 a Server which supports XMPP and can publish its resource(s) to an XMPP server in the Cloud,
475 thus becoming remotely addressable and accessible

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

477 **3.1.19**
478 **Resource**
479 represents an Entity modelled and exposed by the Framework

480 **3.1.20**
481 **Resource Directory**
482 a set of descriptions of resources where the actual resources are held on Servers external to the
483 Device hosting the Resource Directory, allowing lookups to be performed for those resources

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

486 **3.1.21**
487 **Resource Interface**
488 a qualification of the permitted requests on a Resource

489 **3.1.22**
490 **Resource Property**
491 a significant aspect or parameter of a resource, including metadata, that is exposed through the
492 Resource

493 **3.1.23**
494 **Resource Type**
495 a uniquely named definition of a class of Resource Properties and the interactions that are
496 supported by that class

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

498 **3.1.24**
499 **Scene**
500 a static entity that stores a set of defined Resource property values for a collection of Resources

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

503 **3.1.25**

504 **Scene Collection**

505 a collection Resource that contains an enumeration of possible Scene Values and the current
506 Scene Value

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

508 **3.1.26**

509 **Scene Member**

510 a Resource that contains mappings of Scene Values to values of a property in the resource

511 **3.1.27**

512 **Scene Value**

513 a Scene enumerator representing the state in which a Resource can be

514 **3.1.28**

515 **Server**

516 a Device with the role of providing resource state information and facilitating remote interaction
517 with its resources

518 Note 1 to entry: A Server can be implemented to expose non-OCF Device resources to Clients (section 5.5)

519 **3.2 Symbols and abbreviations**

520 **3.2.1**

521 **ACL**

522 Access Control List

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

524 **3.2.2**

525 **CBOR**

526 Concise Binary Object Representation

527 **3.2.3**

528 **CoAP**

529 Constrained Application Protocol

530 **3.2.4**

531 **EXI**

532 Efficient XML Interchange

533 **3.2.5**

534 **IRI**

535 Internationalized Resource Identifiers

536 **3.2.6**

537 **ISP**

538 Internet Service Provider

539 **3.2.7**

540 **JSON**

541 JavaScript Object Notation

542 **3.2.8**

543 **mDNS**

544 Multicast Domain Name Service

545 **3.2.9**
546 **MTU**
547 Maximum Transmission Unit

548 **3.2.10**
549 **NAT**
550 Network Address Translation

551 **3.2.11**
552 **OCF**
553 Open Connectivity Foundation

554 the organization that created this specification

555 **3.2.12**
556 **URI**
557 Uniform Resource Identifier

558 **3.2.13**
559 **URN**
560 Uniform Resource Name

561 **3.2.14**
562 **UTC**
563 Coordinated Universal Time

564 **3.2.15**
565 **UUID**
566 Universal Unique Identifier

567 **3.2.16**
568 **XML**
569 Extensible Markup Language

570 **3.3 Conventions**

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

575 **3.4 Data types**

576 Table 1 contains the definitions of data types used to describe a Resource. The data types are
577 derived from JSON values as defined in ECMA-4-4. However a Resource can overload a JSON
578 defined value to specify a particular subset of the JSON value. These specific data types are
579 defined in Table 1. The data types can be adapted for a particular usage, for example the length
580 of a string can be changed for a specific usage.

581 **Table 1. Data type definition**

Name	JSON value	JSON format value	Description
boolean	false true	n/a	Binary-value {0, 1}.
BSV	string	bsv	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	csv	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	date-time	As defined in ISO 8601. The format is restricted to [yyyy]-[mm]-[dd].
datetime	string	date-time	As defined in ISO 8601.
enum	enum	n/a	Enumerated type.
float	number	float	Signed IEEE 754 single precision float value.
integer	number	integer	Signed 32 bit integer.
json	object/array	n/a	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	n/a	UTF-8 character string shall not exceed a max length of 64 octets unless otherwise specified for a Property value in this specification.
time	string	time	As defined in ISO 8601 but restricted to UTC with a trailing "Z". The format is [hh]:[mm]:[ss]Z.
URI	string	uri	A uniform resource identifier (URI) is a string of characters used to identify a resource according to IETF RFC 3986. The URI value shall not exceed a max length of 256 octets (bytes).
UUID	string	uuid	An identifier formatted according to IETF RFC 4122.

582

583 **4 Document conventions and organization**

584 In this document, features are described as required, recommended, allowed or DEPRECATED as
585 follows:

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

- 587 • These basic features shall be implemented to comply with Core Architecture. The phrases
588 "shall not", and "PROHIBITED" indicate behavior that is prohibited, i.e. that if performed means
589 the implementation is not in compliance.

590 Recommended (or should)(S).

- 591 • These features add functionality supported by Core Architecture and should be implemented.
592 Recommended features take advantage of the capabilities Core Architecture, usually without
593 imposing major increase of complexity. Notice that for compliance testing, if a recommended
594 feature is implemented, it shall meet the specified requirements to be in compliance with these
595 guidelines. Some recommended features could become requirements in the future. The phrase
596 "should not" indicates behavior that is permitted but not recommended.

597 Allowed (may or allowed)(O).

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

601 DEPRECATED.

- 602 • Although these features are still described in this specification, they should not be implemented
603 except for backward compatibility. The occurrence of a deprecated feature during operation of
604 an implementation compliant with the current specification has no effect on the
605 implementation's operation and does not produce any error conditions. Backward compatibility

606 may require that a feature is implemented and functions as specified but it shall never be used
607 by implementations compliant with this specification.

608 Conditionally allowed (CA)

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

611 Conditionally required (CR)

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

615

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

617 Words that are emphasized are printed in italic.

618 **5 Architecture**

619 **5.1 Overview**

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

624 Specifically, the architecture provides:

- 625 • A communication and interoperability framework for multiple market segments (Consumer,
626 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,
627 transports and use cases
- 628 • A common and consistent model for describing the environment and enabling information
629 and semantic interoperability
- 630 • Common communication protocols for discovery and connectivity
- 631 • Common security and identification mechanisms
- 632 • Opportunity for innovation and product differentiation
- 633 • A scalable solution addressing different device capabilities, applicable to smart devices as
634 well as the smallest connected things and wearable devices

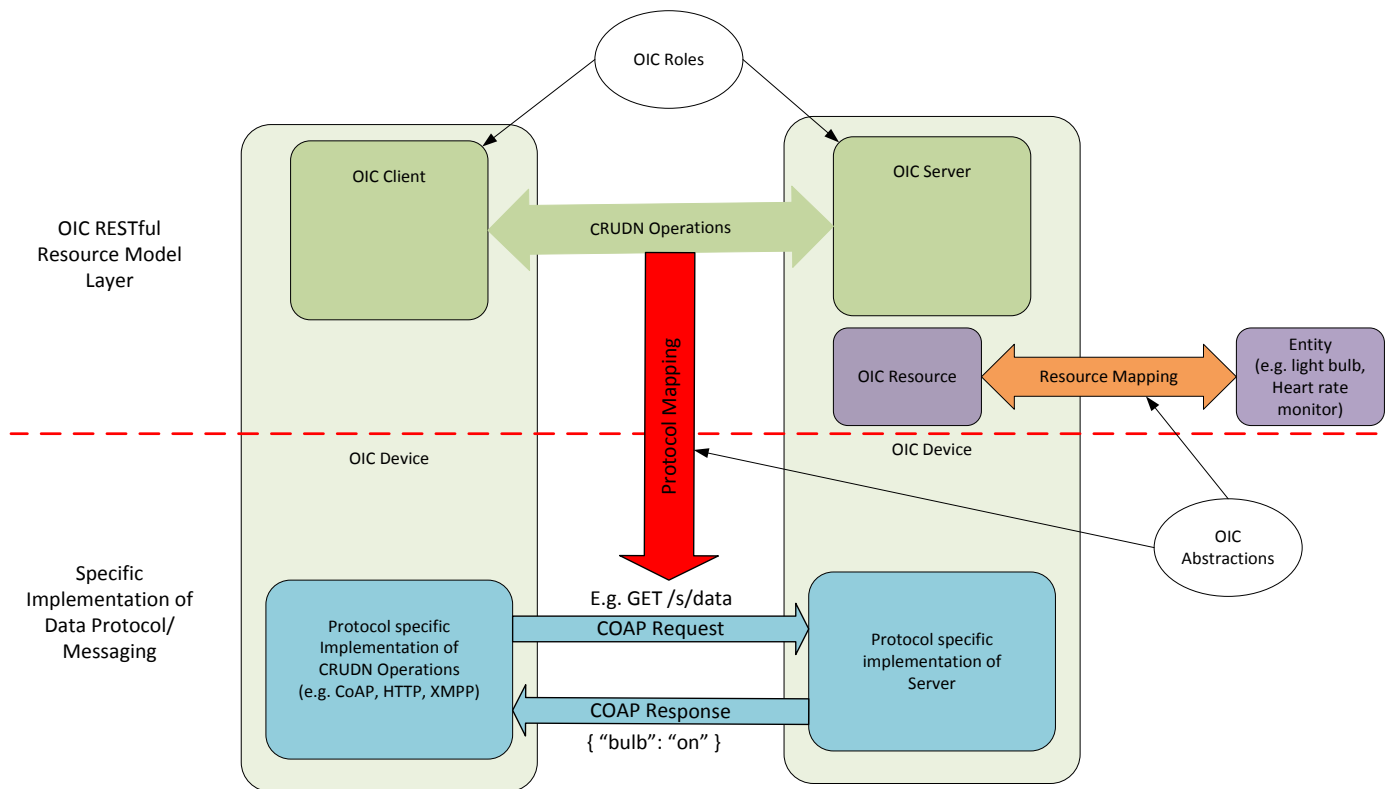
635 The architecture is based on the Resource Oriented Architecture design principles and described
636 in the sections 5.2 through 5.5 respectively. Section 5.2 presents the guiding principles for OCF
637 operations. Section 5.3 defines the functional block diagram and Framework. Section 5.4 provides
638 an example scenario with roles. Section 5.5 provides an example scenario of bridging to non- OCF
639 ecosystem.

640 **5.2 Principle**

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

645 The architecture defines the overall structure of the Framework as an information system and the
 646 interrelationships of the Entities that make up OCF. Entities are exposed as Resources, with their
 647 unique identifiers (URIs) and support interfaces that enable RESTful operations on the Resources.
 648 Every RESTful operation has an initiator of the operation (the client) and a responder to the
 649 operation (the server). In the Framework, the notion of the client and server is realized through
 650 roles (section 5.4). Any Device can act as a Client and initiate a RESTful operation on any Device
 651 acting as a Server. Likewise, any Device that exposes Entities as Resources acts as a Server.
 652 Conformant to the REST architectural style, each RESTful operation contains all the information
 653 necessary to understand the context of the interaction and is driven using a small set of generic
 654 operations, i.e., Create, Read, Update, Delete, Notify (CRUDN) defined in section 8, which include
 655 representations of Resources.

656 Figure 1 depicts the architecture.



657
658

659 **Figure 1: Architecture - concepts**

660

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

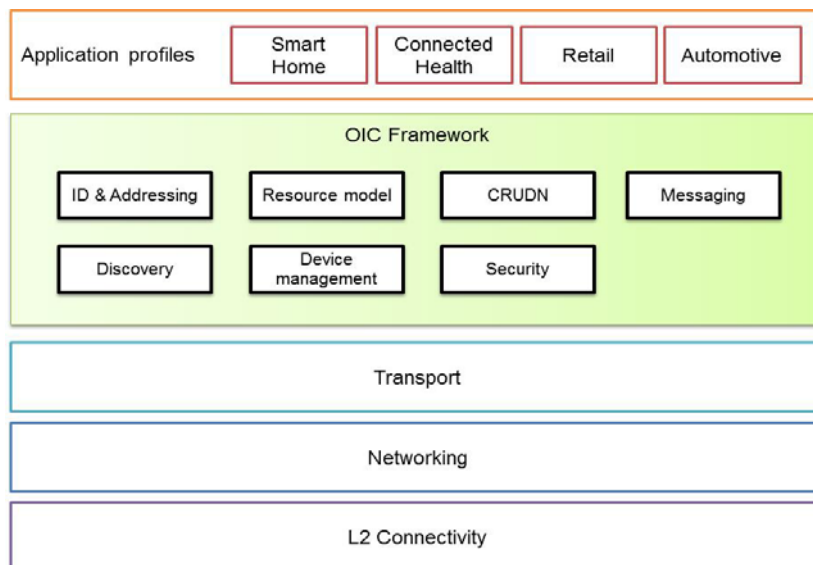
- 663 • Resource model: The resource model provides the abstractions and concepts required to
 664 logically model, and logically operate on the application and its environment. The core resource
 665 model is common and agnostic to any specific application domain such as smart home,
 666 industrial or automotive. For example, the resource model defines a Resource which abstracts
 667 an Entity and the representation of a Resource maps the Entity's state. Other resource model
 668 concepts can be used to model other aspects, for example behavior.

- 669 • RESTful operations: The generic CRUDN operations are defined using the RESTful paradigm
670 to model the interactions with a Resource in a protocol and technology agnostic way. The
671 specific communication or messaging protocols are part of the protocol abstraction and
672 mapping of Resources to specific protocols is provided in section 12.
- 673 • Abstraction: The abstractions in the resource model and the RESTful operations are mapped
674 to concrete elements using abstraction primitives. An entity handler is used to map an Entity
675 to a Resource and connectivity abstraction primitives are used to map logical RESTful
676 operations to data connectivity protocols or technologies. Entity handlers may also be used to
677 map Resources to Entities that are reached over protocols that are not natively supported by
678 OCF.

679

680 5.3 Functional block diagram

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

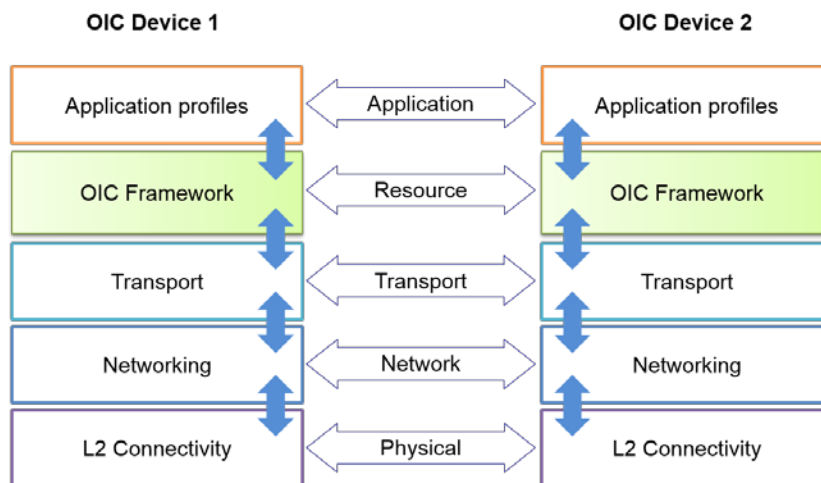


684

685 **Figure 2: Functional block diagram**

- 686 • **L2 connectivity:** Provides the functionalities required for establishing physical and data
687 link layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.
- 688 • **Networking:** Provides functionalities required for Devices to exchange data among
689 themselves over the network (e.g., Internet).
- 690 • **Transport:** Provides end-to-end flow transport with specific QoS constraints. Examples of
691 a transport protocol include TCP and UDP or new Transport protocols under development
692 in the IETF, e.g., Delay Tolerant Networking (DTN).
- 693 • **Framework:** Provides the core functionalities as defined in this specification. The
694 functional block is the source of requests and responses that are the content of the
695 communication between two Devices.
- 696 • **Application profile:** Provides market segment specific data model and functionalities, e.g.,
697 smart home data model and functions for the smart home market segment.

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



700
 701 **Figure 3: Communication layering model**

702 **5.3.1 Framework**

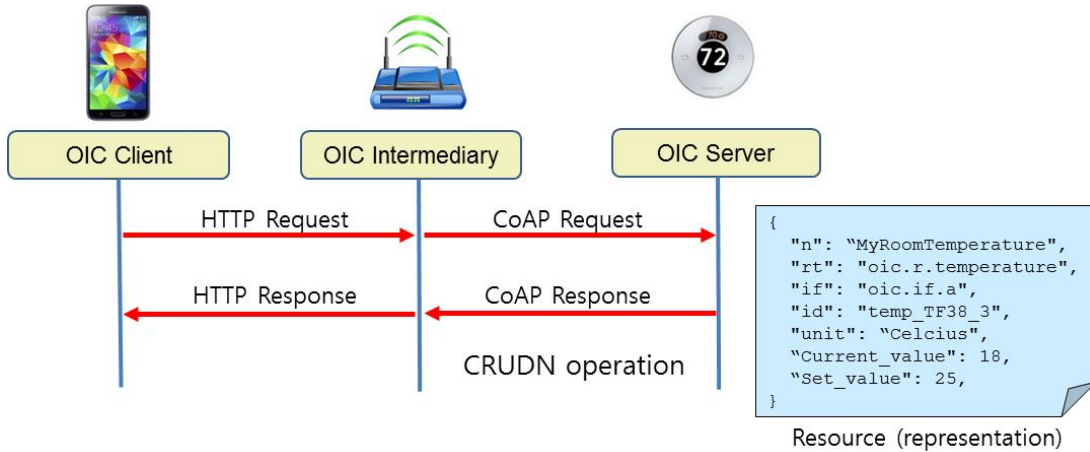
703 Framework consists of functions which provide core functionalities for operation.

- 704 1) **Identification and addressing.** Defines the identifier and addressing capability. The
 705 Identification and addressing function is defined in section 6.
- 706 2) **Discovery.** Defines the process for discovering available
 707 a) Devices (Endpoint Discovery in section 10) and
 708 b) Resources (Resource discovery in section 11.3)
- 709 3) **Resource model.** Specifies the capability for representation of Entities in terms of resources
 710 and defines mechanisms for manipulating the resources. The resource model function is
 711 defined in section 7.
- 712 4) **CRUDN.** Provides a generic scheme for the interactions between a Client and Server as
 713 defined in section 8.
- 714 5) **Messaging.** Provides specific message protocols for RESTful operation, i.e. CRUDN. For
 715 example, CoAP is a primary messaging protocol. The messaging function is defined in section
 716 12.
- 717 6) **Device management.** Specifies the discipline of managing the capabilities of a Device, and
 718 includes device provisioning and initial setup as well as device monitoring and diagnostics.
 719 The device management function is defined in section 11.5.
- 720 7) **Security.** Includes authentication, authorization, and access control mechanisms required for
 721 secure access to Entities. The security function is defined in section 13.

722 **5.4 Example Scenario with roles**

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

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



730

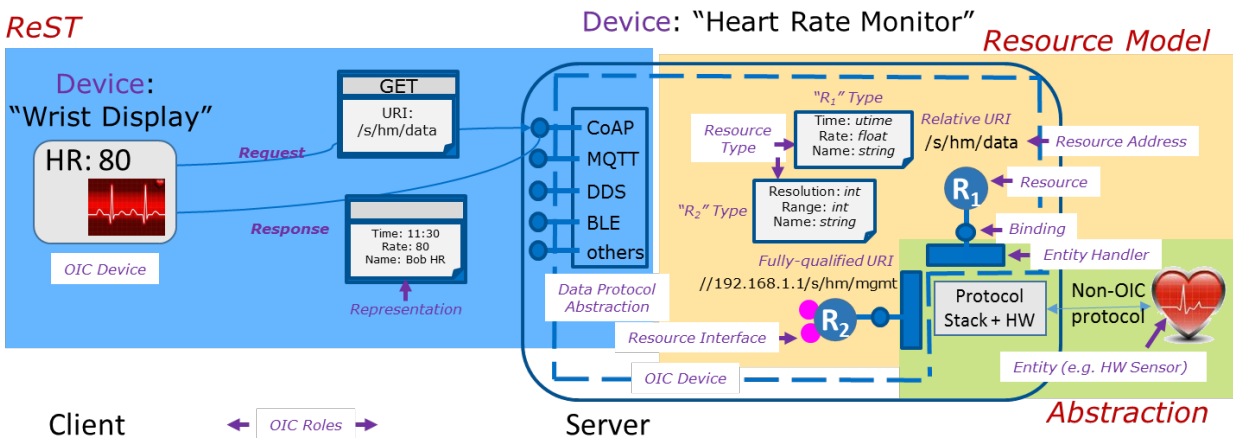
731

Figure 4: Example illustrating the Roles

732 **5.5 Example Scenario: Bridging to Non- OCF ecosystem**

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

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



736

737

Figure 5: Framework - Architecture Detail

738

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



741
742

743

Figure 6: Server bridging to Non- OCF device

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

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

753 **6 Identification and addressing**

754 **6.1 Introduction**

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

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

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

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

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

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

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

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

778 6.2 Identification

779 An identifier shall be unique within the context or domain of use. There are many schemes that
780 may be used to generate an identifier that has the required properties. The identifier may be
781 context-specific in that the identifier is expected to be and guaranteed to be unique only within that
782 context or domain. Identifier may also be context-independent where these identifiers are
783 guaranteed to be unique across all contexts and domains both spatially and temporally. The
784 context-specific identifiers could be defined by simple schemes like monotonic enumeration or may
785 be defined by overloading an address or name, for example an IP address may be an identifier
786 within the private domain behind a gateway in a smart home. On the other hand, context-
787 independent identifiers require a stronger scheme that derives universally unique identities, for
788 example any one of the versions of Universally Unique Identifiers (UUIDs). Context independent
789 identifier may also be generated using hierarchy of domains where the root of the hierarchy is
790 identified with a UUID and sub-domains may generate context independent identifier by
791 concatenating context-specific identifiers for that domain to the context-independent identifier of
792 their parent.

793 6.2.1 Resource identification and addressing

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

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

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

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

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

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

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

806 Each transport binding is responsible for specifying how an OCF URI is converted to a transport
807 protocol URI before sending over the network by the requestor. Similarly on the receiver side, each
808 transport binding is responsible for specifying how to convert from a transport protocol URI to an
809 OCF URI before handing over to the resource model layer on the receiver.

810 If the authority is the local Device, then 'oic' may be used as the authority.

811 The usual form of the authority is

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

- 814 • For IP networks, the hostname or IP address of <authority>
- 815 • For non-IP networks, the name or appropriate identifier.
- 816 • If the <authority> is the Device that hosts the resource then the keyword 'oic' may be used
817 for the <host>.

818 The *path* shall be unique string that unambiguously identifies or references a resource within the
819 context of the Server. In this version of the specification, a path shall not include pct-encoded non-

820 ASCII characters or NUL characters. A *path* shall be preceded by a '/' (slash). The *path* may have
821 '/' (slash) separated segments for human readability reasons. In the OCF context, the '/' (slash)
822 separated segments are treated as a single string that directly references the resources (i.e. a flat
823 structure) and not parsed as a hierarchy. On the Server, the path or some substring in the path
824 may be shortened by using hashing or some other scheme provided the resulting reference is
825 unique within the context of the host.

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

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

831 A URI may be either

- 832 • Fully qualified or
- 833 • Relative

834 *Generation of URI:*

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

840 *Use of URI:*

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

846 **6.3 Namespace:**

847 The relative URI prefix "/oic/" is reserved as a namespace for URIs defined in OCF specifications
848 and shall not be used for URIs that are not defined in OCF specifications.

849 **6.4 Network addressing**

850 The following are the addresses used in this specification:

- 851 • **IP address**

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

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

855 **7 Resource model**

856 **7.1 Introduction**

857 The Resource Model defines concepts and mechanisms that provide consistency and core
858 interoperability between devices in the OCF ecosystems. The Resource Model concepts and
859 mechanisms are then mapped to the transport protocols to enable communication between the

860 devices – each transport provides the communication protocol interoperability. The Resource
861 Model, therefore, allows for interoperability to be defined independent of the transports.

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

867 The primary concepts in the Resource Model are: Entity, Resources, Uniform Resource Identifiers
868 (URI), Resource Types, Properties, Representations, Interfaces, Collections and Links. In addition,
869 the general mechanisms are Create, Update, Retrieve, Delete and Notify. These concepts and
870 mechanisms may be composed in various ways to define the rich semantics and interoperability
871 needed for a diverse set of use cases that the OCF framework is applied to.

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

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

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

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

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

890 The information for Request or Response with the Representation may be communicated "on the
891 wire" by serializing using a transfer protocol or encapsulated in the payload of the transport
892 protocol – the specific method is determined by the normative mapping of the Request or Response
893 to the transport protocol. See section 12 for transport protocols supported.

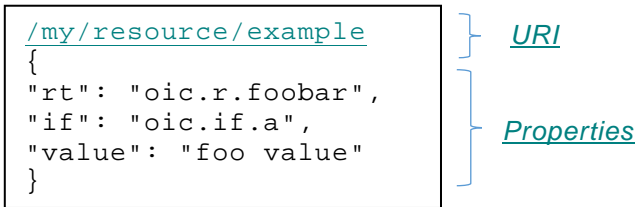
894 The RAML definitions used in this document are normative. This also includes that all defined
895 JSON payloads shall comply with the indicated JSON schema. See Annex D for Resource Types
896 defined in this specification.

897 **7.2 Resource**

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

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

903 specification of the Resource Type.



904

905

Figure 7: Example of a Resource

906

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

911 7.3 Property

912 7.3.1 Introduction

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

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

921 In addition, the Property definition shall have a

- 922 • **Value Type** – the Value Type defines the values that a Property Value may take. The Value
923 Type may be a simple data type (e.g. string, Boolean) as defined in section 3.4 or may be a
924 complex data type defined with a schema. The Value Type may define
 - 925 ○ Value Rules define the rules for the set of values that the Property Value may take.
926 Such rules may define the range of values, the min-max, formulas, set of
927 enumerated values, patterns, conditional values and even dependencies on values
928 of other Properties. The rules may be used to validate the specific values in a
929 Property Value and flag errors.
- 930 • **Mandatory** – specifies if the Property is mandatory or not for a given Resource Type.
- 931 • **Access modes** – specifies whether the Property may be read, written or both. Updates are
932 equivalent to a write. “r” is used for read and “w” is used for write – both may be specified.
933 Write does not automatically imply read.

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

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

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

949 In general, a property is meaningful only within the resource to which it is associated. However a
950 base set of properties that may be supported by all Resources, known as Common Properties,
951 keep their semantics intact across Resources i.e. their “key=value” pair means the same in any
952 Resource. Detailed tables with the above fields for all common properties are defined in section
953 7.3.2.

954 **7.3.2 Common Properties**

955 **7.3.2.1 Introduction**

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

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

971 The following Common Properties for all Resources are specified in section 7.3.2.2 through section
972 7.3.2.6 and summarized as follows:

- 973 • Resource Type (“rt”) – this Property is used to declare the Resource Type of that Resource.
974 Since a Resource could be define by more than one Resource Type the Property Value of the
975 Resource Type Property can be used to declare more than one Resource type. For example:
976 “rt”: [“oic.wk.d”, “oic.d.airConditioner”] declares that the Resource containing this Property is
977 defined by either the “oic.wk.d” Resource Type or the “oic.d.airConditioner” Resource Type.
978 See section 7.3.2.3 for details.
- 979 • Interface (“if”) – this Property declares the Interfaces supported by the Resource. The Property
980 Value of the Interface Property can be multi-valued and lists all the Interfaces supported. See
981 section 7.3.2.4 for details.
- 982 • Name (“n”) – the Property declares “human-readable” name assigned to the Resource. See
983 section 7.3.2.5.
- 984 • Resource Identity (“id”): its Property Value shall be a unique (across the scope of the host
985 Server) instance identifier for a specific instance of the Resource. The encoding of this identifier
986 is device and implementation dependent. See section 7.3.2.6 for details.

987 **7.3.2.2 Property Name and Property Value definitions**

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

- 989 • **Property Name**– the key in "key=value" pair. Property Name is case sensitive and its data type
990 is "string" but only ASCII characters are permitted, and embedded NUL characters are not
991 permitted.
- 992 • **Property Value** – the value in "key=value" pair. Property Value is case sensitive when its data
993 type is "string". Any enum values shall be ASCII only.

994 **7.3.2.3 Resource Type**

995 Resource Type Property is specified in Section 7.4.

996 **7.3.2.4 Interface**

997 Interface Property is specified in Section 7.5.

998 **7.3.2.5 Name**

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

1001 **Table 2. 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)

1002

1003 **7.3.2.6 Resource Identity**

1004 The Resource Identity Property shall be a unique (across the scope of the host Server) instance
1005 identifier for a specific instance of the Resource. The encoding of this identifier is device and
1006 implementation dependent. The Resource Identity Property is as defined in Table 3.

1007 **Table 3. Resource Identity Property Definition**

1008

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

1009

1010 **7.4 Resource Type**

1011 **7.4.1 Introduction**

1012 Resource Type is a class or category of Resources and a Resource is an instance of one or more
1013 Resource Types.

1014 The Resource Types of a Resource is declared using the Resource Type Common Property as
1015 described in Section 7.3.2.3 or in a Link using the Resource Type Parameter.

1016 A Resource Type may either be pre-defined (Core Resource Types in this specification and vertical
 1017 Resource Types in vertical domain specifications) or in custom definitions by manufacturers, end
 1018 users, or developers of Devices (vendor-defined Resource Types). Resource Types and their
 1019 definition details may be communicated out of band (like in documentation) or be defined explicitly
 1020 using a meta-language which may be downloaded and used by APIs or applications. OCF has
 1021 adopted RAML and JSON Schema as the specification method for OCF's RESTful interfaces and
 1022 Resource definitions. OCF defined Interfaces and Resource Types are specified using RAML and
 1023 JSON schema (respectively).

1024 Every Resource Type shall be identified with a Resource Type ID which shall be a lower case
 1025 string with segments separated by a "." (dot). The entire string represents the Resource Type ID.
 1026 When defining the ID each segment may represent any semantics that are appropriate to the
 1027 Resource Type. For example, each segment could represent a namespace. Once the ID has been
 1028 defined, the ID should be used opaquely and an implementations should not infer any information
 1029 from the individual segments. The string "oic", when used as the first segment in the definition
 1030 of the Resource Type ID, is reserved for OCF-defined Resource Types. The Resource Type ID
 1031 may also be a reference to an authority similar to IANA that may be used to find the definition of a
 1032 Resource Type.

1033 7.4.2 Resource Type Property

1034 A Resource when instantiated or created shall have one or more Resource Types that are the
 1035 template for that Resource. The Resource Types that the Resource conforms to shall be declared
 1036 using the "rt" Common Property for the Resource. The Property Value for the "rt" Common Property
 1037 shall be the list of Resource Type IDs for the Resource Types used as templates (i.e., "rt"=<list of
 1038 Resource Type IDs>).

1039 **Table 4. Resource Type Common Property definition**

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

1040 Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and
 1041 the host (i.e. Server) of the Resource.

1042 7.4.3 Resource Type definition

1043 Resource Type is specified as follows:

- 1044 • **Pre-defined URI (optional)** – a pre-defined URI may be specified for a specific Resource Type
 1045 in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that
 1046 Resource Type shall use only the pre-defined URI. An instance of a different Resource Type
 1047 shall not use the pre-defined URI.
- 1048 • **Resource Type Title (optional)** – a human friendly name to designate the resource type.
- 1049 • **Resource Type ID** – the value of "rt" property which identifies the Resource Type, (e.g.,
 1050 oic.wk.p). A lower case string that has segments separated by a '.' (dot); each segment may
 1051 represent a name space and in that case later segments (L -> R) would represent sub-name
 1052 spaces; Implementations shall use these opaquely and use case sensitive string matches.
- 1053 • **Resource Interfaces** – list of the interfaces that may be supported by the resource type.
- 1054 • **Resource Properties** – definition of all the properties that apply to the resource type. The
 1055 resource type definition shall define whether a property is mandatory, conditional mandatory,
 1056 or optional.

- 1057 • **Related Resource Types** (optional) – the specification of other resource types that may be
1058 referenced as part of the resource type, applicable to collections.
- 1059 • **Mime Types** (optional) – mime types supported by the resource including serializations (e.g.,
1060 application/cbor, application/json, application/xml).

1061 Table 5 and Table 6 provide an example description of an illustrative foobar Resource Type and
1062 its associated Properties.

1063 **Table 5. Example foobar Resource Type**

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

1064 **Table 6. Example foobar properties**

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

1065
1066 An instance of the foobar resource type is as shown below

```

1067 {
1068   "rt": "oic.r.foobar",
1069   "if": "oic.if.a",
1070   "value": "foo value"
1071 }

```

1071 An example schema for the foobar resource type is shown below

```

1072 {
1073   "$schema": "http://json-schema.org/draft-04/schema",
1074   "type": "object",
1075   "properties": {
1076     "rt": {"type": "string"},
1077     "if": {"type": "string"},
1078     "value": {"type": "string"}
1079   },
1080   "required": ["rt", "if", "value"]
1081 }

```

1079 7.5 Device Type

1080 A Device Type is a class of Device. Each Device Type defined will include a list of minimum
1081 Resource Types that a device shall implement for that Device Type. A device may expose

1082 additional standard and vendor defined Resource Types beyond the minimum list. The Device
1083 Type is used in Resource discovery as specified in section 11.3.4.

1084 Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in
1085 a Link using the Resource Type Parameter.

1086 A Device Type may either be pre-defined (in vertical domain specifications) or in custom definitions
1087 by manufacturers, end users, or developers of Devices (vendor-defined Device Types). Device
1088 Types and their definition details may be communicated out of band (like in documentation).

1089 Every Device Type shall be identified with a Resource Type ID using the same syntax constraints
1090 as a Resource Type.

1091 **7.6 Interface**

1092 **7.6.1 Introduction**

1093 An Interface provides first a view into the Resource and then defines the requests and responses
1094 permissible on that view of the Resource. So this view provided by an Interface defines the context
1095 for requests and responses on a Resource. Therefore, the same request to a Resource when
1096 targeted to different Interfaces may result in different responses.

1097 An Interface may be defined by either this specification (a Core Interface), the OCF vertical domain
1098 specifications (a “vertical Interface) or manufacturers, end users or developers of Devices (a
1099 “vendor-defined Interface”).

1100 The Interface Property lists all the Interfaces the Resource support. All resources shall have at
1101 least one Interface. The Default Interface shall be defined by an OCF specification and inherited
1102 from the resource type definition. The Default Interface associated with all Resource Types defined
1103 in this specification shall be the supported Interface listed first within the applicable enumeration
1104 in the definition of the Resource Type (see Annex D). All Default Interfaces specified in an OCF
1105 specification shall be mandatory.

1106 In addition to any OCF specification defined interface, all Resources shall support the Baseline
1107 Interface (oic.if.baseline) as defined in section 7.6.3.2.

1108 When an Interface is to be selected for a Request, it shall be specified as query parameter in the
1109 URI of the Resource in the Request message. If no query parameter is specified, then the Default
1110 Interface shall be used. If the selected Interface is not one of the permitted Interfaces on the
1111 Resource then selecting that Interface is an error.

1112 An Interface may accept more than one media type. An Interface may respond with more than one
1113 media type. The accepted media types may be different from the response media types. The media
1114 types are specified with the appropriate header parameters in the transfer protocol. (NOTE: This
1115 feature has to be used judiciously and is allowed to optimize representations on the wire) Each
1116 Interface shall have at least one media type.

1117

1118 **7.6.2 Interface Property**

1119 **Table 7. Resource Interface Property definition**

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

1120 The Interfaces supported by a Resource shall be declared using the Interface Common Property
 1121 (Table 7) as "if=<array of Interfaces>". The Property Value of an Interface Property shall be a
 1122 lower case string with segments separated by a "." (dot). The string "oic", when used as the first
 1123 segment in the Interface Property Value, is reserved for OCF-defined Interfaces. The Interface
 1124 Property Value may also be a reference to an authority similar to IANA that may be used to find
 1125 the definition of an Interface. A Resource Type shall support one or more of the Interfaces defined
 1126 in section 7.6.3.

1127 **7.6.3 Interface methods**

1128 **7.6.3.1 Overview**

1129 The OCF -defined Interfaces are listed in the table below:

1130 **Table 8. OCF standard Interfaces**

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

1131

1132

1133 **7.6.3.2 Baseline Interface**

1134 **7.6.3.2.1 Overview**

1135 The Representation that is visible using the "baseline" Interface includes all the Properties of the
 1136 Resource including the Common Properties. The "baseline" Interface shall be defined for all
 1137 Resource Types. All Resources shall support the "baseline" Interface.

1138 The "baseline" Interface is selected by adding if=oic.if.baseline to the list of query parameters in
 1139 the URI of the target Resource. For example: GET /oic/res?if=oic.if.baseline.

1140 **7.6.3.2.2 Use of RETRIEVE**

1141 The “baseline” Interface is used when a Client wants to retrieve all Properties of a Resource. The
1142 Client includes the URI query parameter definition “?if=oic.if.baseline” in a RETRIEVE request.
1143 When this query parameter definition is included the Server shall respond with a Resource
1144 representation that includes all of the implemented Properties of the Resource. When the Server
1145 is unable to send back the whole Resource representation, it shall reply with an error message.
1146 The Server shall not return a partial Resource representation.

1147 An example response to a RETRIEVE request using the baseline Interface is shown below:

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

1148

1149 **7.6.3.2.3 Use of UPDATE**

1150 Using the baseline Interface, all Properties of a Resource may be modified using an UPDATE
1151 request with a list of Properties and their desired values.

1152 **7.6.3.3 Link List Interface**

1153 **7.6.3.3.1 Overview**

1154 The links list Interface provides a view into the list of Links in a Collection (Resource). The
1155 Representation visible through this Interface has only the Links defined in the Property Value of
1156 the “links” Property – so this Interface is used to manipulate or interact with the list of Links in a
1157 Collection. The Links list may be RETRIEVED using this Interface.

1158 The Interface definition and semantics are given as follows:

- 1159 • The links list Interface name shall be “oic.if.ll”.
- 1160 • If specified in a request (usually in the request header), the serialization in the response shall
1161 be in the format expected in the request.
- 1162 • In response to a RETRIEVE request on the “links list” Interface, the URIs of the referenced
1163 Resources shall be returned as a URI reference.
- 1164 • If there are no links present in a Resource, then an empty list shall be returned.
- 1165 • The Representation determined by this Interface view only includes the Property Value of the
1166 “links” Property.

1167 **7.6.3.3.2 Example: “links list” Interface**

1168 **Example: Request to a Collection**

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

1169

1170 7.6.3.4 Batch Interface

1171 7.6.3.4.1 Overview

1172 The batch Interface is used to interact with a collection of Resources using a single/same Request.
1173 The batch Interface supports methods of Resources in the Links of the Collection, and can be used
1174 to RETRIEVE or UPDATE the Properties of the “linked” Resources with a single Resource
1175 representation.

1176 The batch Interface selects a view into the Links in a Collection – the Request is sent to all the
1177 Links in this view with potential modifications defined in the Parameters of the Link

1178 The batch Interface is defined as follows:

- 1179 • The batch Interface name shall be “oic.if.b”
- 1180 • A Resource with a batch Interface has Links that have Resource references that may be URIs
1181 (fully qualified for remote Resources) or relative references (for local Resources).
- 1182 • If the Link to a Resource does not specify an Interface to use (using the “bp” Link parameter),
1183 then the Request shall be forwarded to the Default Interface of the referenced Resource. If the
1184 “bp” specifies a query using the “q” key then that query shall be used in the query parameter
1185 of the URI formed from the Reference so as to select that Interface in the target Resource.
1186 (See “Link” section for more information on “bp” Parameter)
- 1187 • The original request is modified to create new requests targeting each of the targets in the
1188 Resource Links by substituting the URI in the original request with the URI of the target
1189 Resource in the Link. The payload in the original request is replicated in the payload of the
1190 new Requests.
- 1191 • All the Responses from the “linked” Resources shall be aggregated into single Response to
1192 the Client. The Server may timeout the Response to a time window (if a time window has been
1193 negotiated with the Client then the Server shall not timeout within that window; in the absence
1194 of negotiated window, the Server may choose any appropriate window based on conditions). If
1195 the target Resources cannot process the new request, an empty response or error response
1196 shall be returned. These empty/error Responses shall be included in aggregated Response to
1197 the original Client Request.
- 1198 • The aggregate Response is an array of objects with individual responses. Each response in
1199 the aggregate shall include at least two items: (1) the URI (fully qualified) as “href”: <URI> and
1200 (2) the Representation in the Response declared using the keyword “rep” as the key i.e. “rep”:
1201 { <Representation in individual Response> }.
- 1202 • The Client may choose to restrict the list of Links to which the Request is forwarded by providing
1203 a “filter” in the URI of the Collection to which this original ‘batch’ Interface Request is made.
- 1204 • The Representation in the Link-specific Request may not match the Representation from the
1205 view exposed by the Interface on the target Resource. In such cases, UPDATE using ‘PUT’
1206 method will usually fail and so UPDATE using ‘POST’ method would be appropriate – in this
1207 case the ‘subset’ semantics apply where Properties in the Request which match Properties in
1208 the Resource view exposed shall be modified in the target Resource if the Property is writeable.
- 1209 • A Device that supports the ‘batch’ Interface shall implement both the Client and Server Roles.

1210 7.6.3.4.2 Examples: Batch Interface

1211 Example 1

Resources	<pre>/a/room/1 { "rt": ["acme.room"], "if": ["oic.if.baseline", "oic.if.b"],</pre>
-----------	--

	<pre> "color": "blue", "dimension": "15bx15wx10h", "links": [{"href": "/the/light/1", "rt": ["acme.light"], "if": ["oic.if.a", "oic.if.baseline"], "p":{"bm": 2, "sec": true, "port": 33270}, "ins": 1}, {"href": "/the/light/2", "rt": ["mycorp.light"], if: ["oic.if.a" , "oic.if.baseline"], "p":{"bm": 2, "sec": true, "port": 33270}, "ins": 2}, {"href": "/my/fan/1", "rt": ["hiscorp.fan"], if: ["oic.if.baseline", "oic.if.a"], "p":{"bm": 2, "sec": true, "port": 33270}, "ins": 3 }, {"href": "/his/fan/2", "rt": ["hiscorp.fan"], if: ["oic.if.baseline", "oic.if.a"], "p":{"bm": 2, "sec": true, "port": 33270}, "ins": 4, "bp": {"q": "if=oic.if.a"}}] } /the/light/1 { "rt": ["acme.light"], "if": ["oic.if.a", "oic.if.baseline"], "state": 0, "colourtemp": "2700K" } /the/light/2 { "rt": ["mycorp.light"], "if": ["oic.if.a", "oic.if.baseline"], "state": 1, "color": "red" } /my/fan/1 { "rt": ["hiscorp.fan"], "if": ["oic.if.a", "oic.if.baseline"], "state": 0, "speed": 10 } /his/fan/2 { "rt": ["hiscorp.fan"], "if": ["oic.if.a", "oic.if.baseline"], "state": 0, "speed": 20 } </pre>
<p>Use of batch</p>	<p>Request: GET /a/room/1?if=oic.if.b</p> <p>Becomes the following individual responses issued by the Device in the Client role</p> <p>GET /the/light/1 (NOTE: Uses the default Interface: 'sensor')</p>

	<pre>GET /the/light/2 (NOTE: Uses the default Interface: 'sensor')</pre> <pre>GET /my/fan/1 (NOTE: Uses the default Interface: 'baseline')</pre> <pre>GET /his/fan/2?if=oic.if.a (NOTE: Interface from "bp" Link parameter: 'actuator')</pre> <p>Response:</p> <pre>[{ "href": "oic://<devID>/the/light/1", "rep": { "state": 0, "colortemp": "2700K" } }, { "href": "oic://<devID>/the/light/2", "rep": { "state": 1, "color": "red" } }, { "href": "oic://<devID>/my/fan/1", "rep": { "rt": ["hiscorp.fan"], "if": ["oic.if.a", "oic.if.baseline"], "state": 0, "speed": "10" } }, { "href": "oic://<devID>/his/fan/2", "rep": { "state": 0, "speed": "20" } }]</pre>
<p>Use of batch</p> <p>(UPDATE has POST semantics)</p>	<pre>UPDATE /a/room/1?if=oic.if.b { "state": 1 }</pre> <p>becomes</p> <pre>UPDATE /the/light/1 { "state": 1 } UPDATE /my/fan/1 { "state": 1 } UPDATE /his/fan/2?if=oic.if.a { "state": 1 }</pre> <p>This turns on all the lights (except /the/light/1 Resource) and fans on in the room since all the Resources have "state" as a Property. /the/light/1 has the 'sensor' interface as default and so POST is not supported for 'sensor' Interface (the Device hosting /a/room/1 does not send this Request)</p>
<p>Use of batch</p> <p>(UPDATE has POST semantics)</p>	<pre>UPDATE /a/room/1?if=oic.if.b { "state": 1, "color": "blue" }</pre> <p>This turns on all the lights (except /the/light/1 Resource) and fans in the room but also sets the color of /the/light/2 to "blue"</p>

1212

1213 Example that further shows the "links list" and "batch" interface

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

1214

1215 **7.6.3.5 Actuator Interface**

1216 The actuator Interface is the Interface for viewing Resources that may be actuated i.e. changes
1217 some value within or the state of the entity abstracted by the Resource:

- 1218
- The actuator Interface name shall be “oic.if.a”
 - The actuator Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable JSON; the actuator interface may also expose in the Resource Representation optional Properties as defined by the applicable JSON schema that are implemented by the target Device.
- 1219
1220
1221
1222

For the following Resource

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

```

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

```

```
}

```

1223

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

1224

NOTE: The example here is with respect to Figure 8

1. Retrieving values of an actuator

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

Response:

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

2. Correct use of actuator:

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

```
{
  "settemp": 20
}
```

Response:

```
{
  Ok
}
```

3. Incorrect use of actuator

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

```
{
  "if": "oic.if.s"    ← this is visible through baseline
}
```

Interface

```
{
}
```

Response:

```
{
  Error
}
```

1225

Figure 9: Example - Actuator Interface

1226

- A RETRIEVE request using this Interface shall return the Representation for this Resource subject to any query and filter parameters that may also exist

1227

1228

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

1229

1230 7.6.3.6 Sensor Interface

1231

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

1232

1233

- The sensor Interface name shall be "oic.if.s"

1234

- The sensor Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable JSON; the sensor interface may also expose in the Resource

1235

- 1236 Representation optional Properties as defined by the applicable JSON schema that are
1237 implemented by the target Device.
- 1238 • A RETRIEVE request using this Interface shall return this Representation for the Resource
1239 subject to any query and filter parameters that may also exist
 - 1240 •

NOTE: The example here is with respect to Figure 8

1. Retrieving values of sensor

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

```
Response: {
  "currenttemp": 7
}
```

2. Incorrect use of sensor

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

3. Incorrect use of sensor

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

- 1241
- 1242 **7.6.3.7 Read-only Interface**
- 1243 The read-only Interface exposes only the Properties that may be “read”. This includes Properties
1244 that may be “read-only”, “read-write” but not Properties that are “write-only” or “set-only”. The
1245 applicable methods that can be applied to a Resource is RETRIEVE only. An attempt by a Client
1246 to apply a method other than RETRIEVE to a Resource shall be rejected with an error response
1247 code.
- 1248 **7.6.3.8 Read-write Interface**
- 1249 The read-write Interface exposes only the Properties that may be “read” and “written”. The “read-
1250 only” Properties shall not be included in Representation for the “read-write” Interface. This is a
1251 generic Interface to support “reading” and “setting” Properties in a Resource. The applicable
1252 methods that can be applied to a Resource are RETRIEVE and UPDATE only. An attempt by a

1253 Client to apply a method other than RETRIEVE or UPDATE to a Resource shall be rejected with
1254 an error response code.

1255 **7.7 Resource representation**

1256 Resource representation captures the state of a Resource at a particular time. The resource
1257 representation is exchanged in the request and response interactions with a Resource. A Resource
1258 representation may be used to retrieve or update the state of a resource.

1259 The resource representation shall not be manipulated by the data connectivity protocols and
1260 technologies (e.g., CoAP, UDP/IP or BLE).

1261 **7.8 Structure**

1262 **7.8.1 Introduction**

1263 In many scenarios and contexts, the Resources may have either an implicit or explicit structure
1264 between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The
1265 Framework provides the means to model and map these structures and the relationships among
1266 Resources. The primary building block for resource structures in Framework is the collection. A
1267 collection represents a container, which is extensible to model complex structures.

1268 **7.8.2 Resource Relationships**

1269 Resource relationships are expressed as Links. A Link embraces and extends typed web links
1270 concept as a means of expressing relationships between Resources. A Link consists of a set of
1271 Parameters that define:

- 1272 • a context URI,
- 1273 • a target URI,
- 1274 • a relation from the context URI to the target URI
- 1275 • elements that provide metadata about the target URI, the relationship or the context of the Link.

1276 The target URI is mandatory and the other items in a Link are optional. Additional items in the Link
1277 may be made mandatory based on the use of the links in different contexts (e.g. in collections, in
1278 discovery, in bridging etc.). Schema for the Link payload is provided in Annex D.

1279 An example of a Link is shown in

```
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "/room2"oic.if.baseline"], "p":  
{"bm": 3, "sec": true, "port": 33275}, "rel": "contains" }
```

1280 **Figure 10: Example of a Link**

1281 Two Links are distinct from each other when at least one parameter is different. For example the
1282 two Links shown in Figure 11 are distinct and can appear in the same list of Links.

```
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm":  
2, "sec": true, "port": 33275}, "rel": "contains" }  
  
{ "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p": {"bm":  
2, "sec": true, "port": 33275}, "rel": "activates" }
```

1283 **Figure 11: Example of distinct Links**

1284 The specification may mandate Parameters and Parameter values as required for certain
1285 capabilities. For all Links returned in a response to a RETRIEVE on /oic/res, if a Link does not
1286 explicitly include the “rel” Parameter, a value of “rel”=“hosts” shall be assumed . The relation value
1287 of “hosts” is defined by IETF RFC 6690 and registered in the IANA Registry for Link Relations at
1288 [\[http://www.iana.org/assignments/link-relations/link-relations.xhtml\]](http://www.iana.org/assignments/link-relations/link-relations.xhtml)

1289 As shown in D.2.8 the relation between the context URI and target URI in a Link is specified using
1290 the “rel” JSON element and the value of this element specifies the particular relation.

1291 The context URI of the Link shall implicitly be the URI of the Resource (or specifically a Collection)
1292 that contains the Link unless the Link specifies the anchor parameter. The anchor parameter is
1293 used to change the context URI of a Link – the relationship with the target URI is based off the
1294 anchor URI when the anchor is specified. An example of using anchors in the context of Collections
1295 – a floor has rooms and rooms have lights – the lights may be defined in floor as Links but the
1296 Links will have the anchor set to the URI of the rooms that contain the lights (the relation is
1297 contains). This allows all lights in a floor to be turned on or off together while still having the lights
1298 defined with respect to the rooms that contain them (lights may also be turned on by using the
1299 room URI too).

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

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

1300 **Figure 12: Example of use of anchor in Link**

1301 **7.8.2.1 Parameters**

1302 **7.8.2.1.1 “ins” or Link Instance Parameter**

1303 The “ins” parameter identifies a particular Link instance in a list of Links. The "ins" parameter may
1304 be used to modify or delete a specific Link in a list of Links. The value of the “ins” parameter is set
1305 at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has
1306 been set, the “ins” parameter shall not be modified for as long as the Link is a member of that list.

1307 **7.8.2.1.2 “p” or Policy Parameter**

1308 The Policy Parameter defines various rules for correctly accessing a Resource referenced by a
1309 target URI. The Policy rules are configured by a set of key-value pairs as defined below.

1310 The policy Parameter "p" is defined by:

- 1311 • “bm” key: The “bm” key corresponds to an integer value that is interpreted as an 8-bit bitmask.
 1312 Each bit in the bitmask corresponds to a specific Policy rule. The following rules are specified
 1313 for “bm”:
 1314

Bit Position	Policy rule	Comment
Bit 0 (the LSB)	discoverable	<p>The discoverable rule defines whether the Link is to be included in the Resource discovery message via /oic/res.</p> <ul style="list-style-type: none"> • If the Link is to be included in the Resource discovery message, then “p” shall include the “bm” key and set the discoverable bit to value 1. • If the Link is NOT to be included in the Resource discovery message, then “p” shall either include the “bm” key and set the discoverable bit to value 0 or omit the “bm” key entirely.
Bit 1 (2 nd LSB)	observable	<p>The observable rule defines whether the Resource referenced by the target URI supports the NOTIFY operation.</p> <ul style="list-style-type: none"> • If the Resource supports the NOTIFY operation, then “p” shall include the “bm” key and set the observable bit to value 1. • If the Resource does NOT support the NOTIFY operation, then “p” shall either include the “bm” key and set the observable bit to value 0 or omit the “bm” key entirely.
Bits 2-7	--	Reserved for future use. All reserved bits in “bm” shall be set to value 0.

1315
 1316 Note that if all the bits in “bm” are defined to value 0, then the “bm” key may be omitted entirely
 1317 from “p” as an efficiency measure. However, if any bit is set to value 1, then “bm” shall be
 1318 included in “p” and all the bits shall be defined appropriately.

- 1319 • “sec” key: The “sec” key corresponds to a Boolean value that indicates whether the Resource
 1320 referenced by the target URI is accessed via an encrypted connection. If “sec” is true, the
 1321 resource is accessed via an encrypted connection, using the “port” specified (see below). If
 1322 “sec” is false, the resource is accessed via an unencrypted connection, or via an encrypted
 1323 connection (if such a connection is made using the “port” settings for another Resource, for
 1324 which “sec” is true).

- 1325 • “port” key: The “port” key corresponds to an integer value that is used to indicate the port
 1326 number where the Resource referenced by the target URI may be accessed via an encrypted
 1327 connection.

- 1328 • If the Resource is only available via an encrypted connection (i.e. DTLS over IP), then
 1329 o “p” shall include the “sec” key and its value shall be true.
 1330 o “p” shall include the “port” key and its value shall be the port number where the
 1331 encrypted connection may be established.

- 1332 • If the Resource is not available via an encrypted connection, then
 - 1333 ○ "p" shall include the "sec" key and its value shall be false or "p" shall omit the "sec"
 - 1334 key; the default value of "sec" is false.
 - 1335 ○ "p" shall omit the "port" key.
 - 1336 ○ A Resource that is available via either an encrypted or unencrypted connection
 - 1337 follows the population scheme defined in this clause.
- 1338 • Access to the Resource on the port specified by the "port" key shall be made by an encrypted
- 1339 connection (e.g. coaps://). (Note that unencrypted connection to the Resource may be possible
- 1340 on a separate port discovered thru multicast discovery).
- 1341 • Note that access to the Resource is controlled by the ACL for the Resource. A successful
- 1342 encrypted connection does not ensure that the requested action will succeed. See
- 1343 OCF Security – Access Control section for more information.

1344 Example 1: below shows the Policy Parameter for a Resource that is discoverable but not

1345 observable, and for which authenticated accesses shall be done via CoAPS port 33275::

```

"p": { "bm": 2, "sec": true,
      "port": 33275 }
```

1348 7.8.2.1.3 “type” or Media Type Parameter

1349 The “type” Parameter may be used to specify the various media types that are supported by a

1350 specific target Resource. The default type of "application/cbor" shall be used when the “type”

1351 element is omitted. Once a Client discovers this information for each Resource, it may use one of

1352 the available representations in the appropriate header field of the Request or Response.

1353 7.8.2.1.4 “bp” or the Batch Interface Parameter

1354 The “batch” Parameter "bp" is used to specify the modifications to the target URI as the "batch"

1355 Request is forwarded through this Link. The "q" element in the value defines the query string that

1356 shall be appended to the "href" to make the target URI. The "q" query string may contain Property

1357 strings that are valid in that context. For example: Given a Collection as follows

```

/room2
{
  "if": "oic.if.b",
  "colour": "blue",
  "links": [
    { "href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a", "oic.if.baseline"], "p":
      {"bm": 2, "sec": true, "port": 33277}, "rel": "contains", "bp": { "q": "if=oic.if.baseline" } }
  ]
}
```

1358 The following is the sequence for batch request to /room2

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

1359 See the Interfaces section 7.5 for more details on the "batch" Interface.

1360 **7.8.2.1.5 "di" or Device ID parameter**

1361 The "di" Parameter specifies the device ID of the Device that hosts the target Resource defined in
1362 the in the "href" Parameter.

1363 The device ID may be used to qualify a relative reference used in the "href" or to lookup endpoint
1364 information for the relative reference.

1365 **7.8.2.1.6 "buri" or base URI Parameter**

1366 The "buri" Parameter is the base URI to which the relative reference in "href" is resolved to. The
1367 base URI and relative reference may be used to construct the URI to the target for the Link. The
1368 base URI shall use the OCF Scheme for the URI defined in section 6.

1369 **7.8.2.2 Formatting**

1370 When formatting in JSON, the list of Links shall be an array. The first element of the array shall be
1371 a JSON object called the "tags block". This object may be empty or have keys that are the
1372 Parameters from the list of Parameters for the Link. The "href" parameter shall not appear in the
1373 "tags block". The second element of this array shall be a list of Links.

1374 For each list of Links the Parameters that appear in the "tags block" shall apply to each of the links
1375 in the list of Links array associated with this tags block.

1376 A null list of Links shall have a null "tags block" and both shall not be included.

1377 NOTE: By this organization the list of Links is recursive and the "tags block" allows for a compact representation where
1378 Parameters shared by multiple Links don't need to be repeated in each Links and can be factored into the "tags block".

1379 For example a list of Links with "tags" block.

```
[
  {
    "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
  },
  [
    {
      "href": "/oic/d",
      "rt": ["oic.d.light", "oic.wk.d"],
      "if": [ "oic.if.r", "oic.if.baseline" ],
      "p": {"bm": 1, "sec": true, "port": 33854}
    },
    {
      "href": "/oic/p",
      "rt": ["oic.wk.p"],
      "if": [ "oic.if.r", "oic.if.baseline" ],
      "p": {"bm": 1, "sec": true, "port": 33854}
    },
    {
      "href": "/switch",
      "rt": ["oic.r.switch.binary"],
      "if": [ "oic.if.a", "oic.if.baseline" ],
      "p": {"bm": 3, "sec": true, "port": 33854},
      "mt": [ "application/cbor", "application/exi+xml" ]
    },
    {
      "href": "/brightness",
      "rt": [ "oic.r.light.brightness" ],
      "if": [ "oic.if.a", "oic.if.baseline" ],
    }
  ]
]
```

```

    "p": { "bm": 3, "sec": true, "port": 33854 }
  }
]

```

1380

Figure 13: Example “list of Links”

1381 **7.8.2.3 List of Links in a Collection**

1382 A list of Links in a Resource shall be included in that Resource as the value of the “links” Property
 1383 of that Resource. A Resource that contains Links is a Collection.

1384 A Resource with a list of Links

```

/Room1
{
  "rt": "my.room",
  "if": [ "oic.if.ll", "oic.if.baseline" ],
  "color": "blue"
  "links":
  [
    {
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
    },
    [
      {
        "href": "/oic/d",
        "rt": [ "oic.d.light", "oic.wk.d" ],
        "if": [ "oic.if.r", "oic.if.baseline" ],
        "p": { "bm": 1, "sec": true, "port": 33822 }
      },
      {
        "href": "/oic/p",
        "rt": [ "oic.wk.p" ],
        "if": [ "oic.if.r", "oic.if.baseline" ],
        "p": { "bm": 1, "sec": true, "port": 33822 }
      },
      {
        "href": "/switch",
        "rt": [ "oic.r.switch.binary" ],
        "if": [ "oic.if.a", "oic.if.baseline" ],
        "p": { "bm": 3, "sec": true, "port": 33822 },
        "mt": [ "application/cbor", "application/exi+xml" ]
      },
      {
        "href": "/brightness",
        "rt": [ "oic.r.light.brightness" ],
        "if": [ "oic.if.a", "oic.if.baseline" ],
        "p": { "bm": 3, "sec": true, "port": 33822 }
      }
    ]
  ]
}

```

1385

Figure 14: List of Links in a Resource

1386 **7.8.2.4 Usage Cases – Resource discovery**

1387 The OCF architecture utilizes typed Links as a mechanism for bootstrapping Resource discovery
1388 through the known Core Resource /oic/res. A RETRIEVE operation on /oic/res returns (among
1389 other things) a serialized representation of typed Links to Resources that are discoverable on that
1390 Device.

1391 The serialization format should be negotiated using the underlying transport protocol (i.e. using
1392 Accept and Content-Type headers in case of CoAP). By default, OCF uses CBOR as the payload.
1393 The payload (content) in CBOR for Links is described with the JSON Schema in D.2.8. Other
1394 serializations (e.g. XML/EXI) may be defined in future versions of this specification. The JSON
1395 Schema that specifies the representation of the response to /oic/res is defined D.8.

1396 **7.8.3 Collections**

1397 **7.8.3.1 Overview**

1398 A Resource that contains one or more references (specified as Links) to other resources is an
1399 Collection. These reference may be related to each other or just be a list; the Collection provides
1400 a means to refer to this set of references with a single handle (i.e. the URI). A simple resource is
1401 kept distinct from a collection. Any Resource may be turned into an Collection by binding resource
1402 references as Links. Collections may be used for creating, defining or specifying hierarchies,
1403 indexes, groups, and so on.

1404 A Collection shall have at least one Resource Type and at least one Interface bound at all times
1405 during its lifetime. During creation time of a collection the resource type and interfaces are
1406 specified. The initial defined resource types and interfaces may be updated during its life time.
1407 These initial values may be overridden using mechanism used for overriding in the case of a
1408 Resource. Additional resource types and interfaces may be bound to the Collection at creation or
1409 later during the lifecycle of the Collection.

1410 A Collection shall define the “links” Common Property. The value of the “links” Property is an array
1411 with zero or more Links. The target URIs in the Links may reference another Collection or another
1412 Resource. The referenced Collection or Resource may reside on the same Device as the Collection
1413 that includes that Link (called a local reference) or may reside on another Device (called a remote
1414 reference). The context URI of the Links in the “links” array shall (implicitly) be the Collection that
1415 contains that “links” property. The (implicit) context URI may be overridden with explicit
1416 specification of the “anchor” parameter in the Link where the value of “anchor” is the new base of
1417 the Link.

1418 A Resource may be referenced in more than one Collection, therefore, a unique parent-child
1419 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the
1420 Resource referenced in the Collection, i.e., the application may use Collections to represent a
1421 relationship but none is automatically implied or defined. The lifecycles of the Collection and the
1422 referenced Resource are also independent of one another.

1423 If the “drel” property is defined for the Collection then all Links that don’t explicitly specify a
1424 relationship shall inherit this default relationship in the context of that Collection. The default
1425 relationship defines the implicit relationship between the Collection and the target URI in the Link.

1426 The list of Links defined in a Collection may be either a simple list of Links as illustrated in Figure
1427 16 or may be a list of tagged Links sets as illustrated in Figure 17. For the former, the value of the
1428 “links” Property is a simple array of Links. For the later, the value of the “links” Property is an array
1429 where each element is a resource containing a Links array and a set of one or more key-value
1430 pairs; the key-value pairs are the tags for the Links array (the key is the tag name and the value
1431 is the tag value)

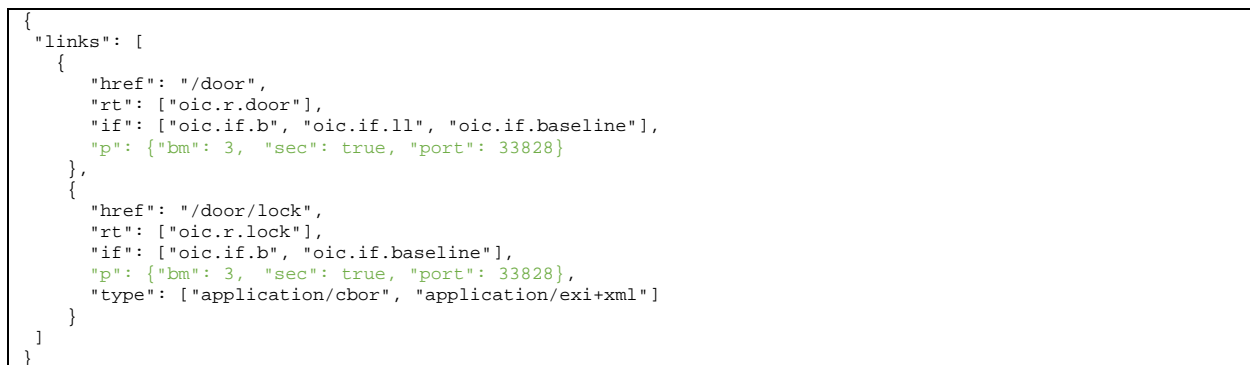


1432

1433

Figure 15: Example showing parts of Collection and Links

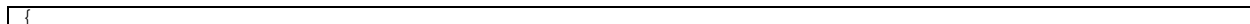
1434



1435

Figure 16: Example Collection with simple links (JSON)

1436



```

"links": [
  [
    {
      "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
    },
    [
      {
        "href": "/door",
        "rt": ["oic.r.door"],
        "if": ["oic.if.b", "oic.if.ll", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 32807}
      },
      {
        "href": "/door/lock",
        "rt": ["oic.r.lock"],
        "if": ["oic.if.b", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 32807},
        "type": ["application/cbor", "application/exi+xml"]
      }
    ]
  ],
  [
    {
      "di": "08854960-736F-46F7-BEC2-9E6CBD61BDC9"
    },
    [
      {
        "href": "/light",
        "rt": ["oic.r.light"],
        "if": ["oic.if.s", "oic.if.baseline"],
        "p": {"bm": 3, "sec": true, "port": 32808}
      }
    ]
  ],
  [
    {
      "href": "/binarySwitch",
      "rt": ["oic.r.switch.binary"],
      "if": ["oic.if.a", "oic.if.baseline"],
      "p": {"bm": 3, "sec": true, "port": 32808},
      "type": ["application/cbor"]
    }
  ]
]
]
}

```

Figure 17: Example Collection with tagged Links (JSON)

1437

1438 Note: Example shows only one tag; each tag has the same tag name, i.e., "di", but have different tag values.

1439

1440 A Collection may be:

- 1441 • A pre-defined Collection where the Collection has been defined a priori and the Collection is
- 1442 static over its lifetime. Such Collections may be used to model, for example, an appliance that
- 1443 is composed of other devices or fixed set of resource representing fixed functions.
- 1444 • A Device local Collection where the Collection is used only on the Device that hosts the
- 1445 Collection. Such collections may be used as a short-hand on a client for referring to many
- 1446 Servers as one.
- 1447 • A centralized Collection where the Collection is hosted on an Device but other Devices may
- 1448 access or update the Collection
- 1449 • A hosted Collection where the collection is centralized but is managed by an authorized agent
- 1450 or party.

1451 **7.8.3.2 Collection Properties**

1452 An Collection shall define the "links" Property. In addition, other Properties may be defined for the

1453 Collection by the Resource Type. The mandatory and recommended Common Properties for

1454 Collection are shown in Table 9. This list of Common Properties are in addition to those defined

1455 for Resources in section 7.3.2. When a property is repeated in Table 9 , the conditions in this

1456 definition shall override those in the general list for Resources.

1457
1458

Table 9: Common Properties for Collections (in addition to Common Properties defined in section 7.3.2)

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

1459

1460 The Properties of a Collection may not be modified.

1461 **7.8.3.3 Default resource type**

1462 A default Resource Type, oic.wk.col, shall be available for Collections. This Resource Type shall
1463 be used only when another type has not been defined on the Collection or when no Resource Type
1464 has been specified at the creation of the Collection.

1465 The default Resource Type provides support for the Common Properties including the "links"
1466 Property. For the default resource type, the value of "links" shall be a simple array of Links and
1467 tagging of links shall not be supported.

1468 The default Resource Type shall support the 'baseline' and 'links list' Interfaces. The default
1469 Interface shall be the 'links list' Interface.

1470 **8 CRUDN**

1471 **8.1 Overview**

1472 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for
1473 manipulating Resources. These operations are performed by a Client on the resources contained
1474 in an Server.

1475 On reception of a valid CRUDN operation an Server hosting the Resource that is the target of the
1476 request shall generate a response depending on the Interface included in the request; or based
1477 on the Default Interface for the Resource Type if no Interface is included.

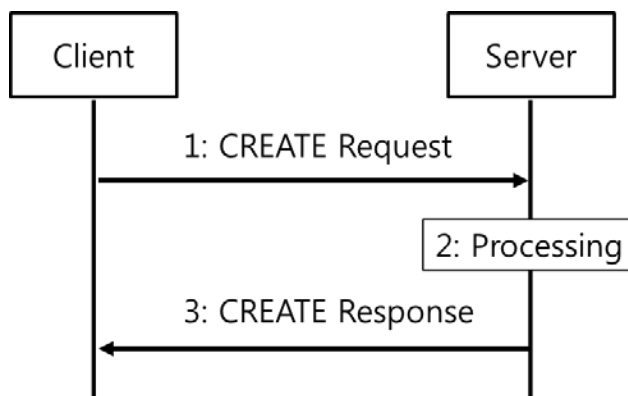
1478 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in
 1479 Table 10. A Device shall use CBOR as the default payload (content) encoding scheme for resource
 1480 representations included in CRUDN operations and operation responses; a Device may negotiate
 1481 a different payload encoding scheme (e.g, see in section 12.2.4 for CoAP messaging). The
 1482 following subsections specify the CRUDN operations and use of the parameters. The type
 1483 definitions for these terms will be mapped in the messaging section for each protocol.

1484 **Table 10. Parameters of CRUDN messages**

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

1485 **8.2 CREATE**

1486 The CREATE operation is used to request the creation of new Resources on the Server. The
 1487 CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 18
 1488 and described below.



1489

1490

Figure 18. CREATE operation

1491 **8.2.1 CREATE request**

1492 The CREATE request message is transmitted by the Client to the Server to create a new Resource
1493 by the Server. The CREATE request message will carry the following parameters:

- 1494 • *fr*: Unique identifier of the Client
- 1495 • *to*: URI of the target resource responsible for creation of the new resource.
- 1496 • *ri*: Identifier of the CREATE request
- 1497 • *cn*: Information of the resource to be created by the Server
 - 1498 i) *cn* will include the URI and resource type property of the resource to be created.
 - 1499 ii) *cn* may include additional properties of the resource to be created.
- 1500 • *op*: CREATE

1501 **8.2.2 Processing by the Server**

1502 Following the receipt of a CREATE request, the Server may validate if the Client has the
1503 appropriate rights for creating the requested resource. If the validation is successful, the Server
1504 creates the requested resource. The Server caches the value of *ri* parameter in the CREATE
1505 request for inclusion in the CREATE response message.

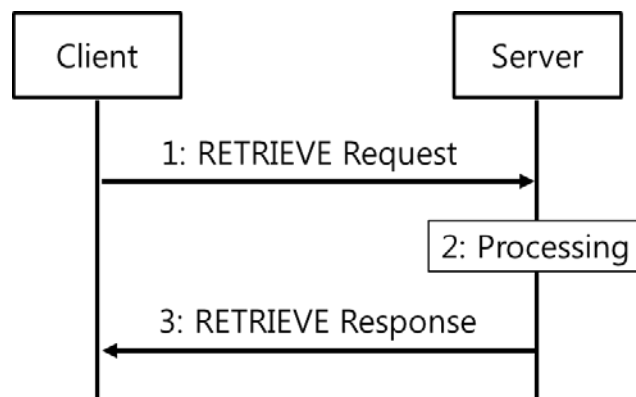
1506 **8.2.3 CREATE response**

1507 The Server shall transmit a CREATE response message in response to a CREATE request
1508 message from a Client. The CREATE response message will include the following parameters.

- 1509 • *fr*: Unique identifier of the Server
- 1510 • *to*: Unique identifier of the Client
- 1511 • *ri*: Identifier included in the CREATE request
- 1512 • *cn*: Information of the resource as created by the Server.
 - 1513 i) *cn* will include the URI of the created resource.
 - 1514 ii) *cn* will include the resource representation of the created resource.
- 1515 • *rs*: The result of the CREATE operation

1516 **8.3 RETRIEVE**

1517 The RETRIEVE operation is used to request the current state or representation of a Resource.
1518 The RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in
1519 Figure 19 and described below.



1520

1521

Figure 19. RETRIEVE operation

1522 8.3.1 RETRIEVE request

1523 RETRIEVE request message is transmitted by the Client to the Server to request the
1524 representation of a Resource from an Server. The RETRIEVE request message will carry the
1525 following parameters.

- 1526 • *fr*: Unique identifier of the Client
- 1527 • *to*: URI of the resource the Client is targeting
- 1528 • *ri*: Identifier of the RETRIEVE request
- 1529 • *op*: RETRIEVE

1530 8.3.2 Processing by the Server

1531 Following the receipt of a RETRIEVE request, the Server may validate if the Client has the
1532 appropriate rights for retrieving the requested data and the properties are readable. The Server
1533 caches the value of *ri* parameter in the RETRIEVE request for use in the response.

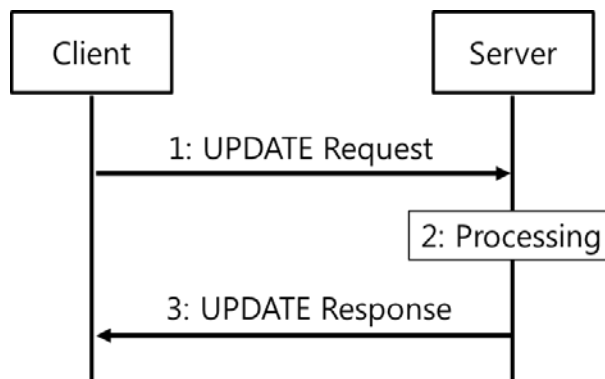
1534 8.3.3 RETRIEVE response

1535 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
1536 message from a Client. The RETRIEVE response message will include the following parameters.

- 1537 • *fr*: Unique identifier of the Server
- 1538 • *to*: Unique identifier of the Client
- 1539 • *ri*: Identifier included in the RETRIEVE request
- 1540 • *cn*: Information of the resource as requested by the Client
 - 1541 i) *cn* should include the URI of the resource targeted in the RETRIEVE request
- 1542
- 1543 • *rs*: The result of the RETRIEVE operation

1544 8.4 UPDATE

1545 The UPDATE operation is either a Partial UPDATE or a complete replacement of the information
1546 in a Resource in conjunction with the interface that is also applied to the operation. The UPDATE
1547 operation is initiated by the Client and consists of three steps, as depicted in Figure 20 and
1548 described below.



1549

1550 Figure 20. UPDATE operation

1551 **8.4.1 UPDATE request**

1552 The UPDATE request message is transmitted by the Client to the Server to request the update of
1553 information of a Resource on the Server. The UPDATE request message will carry the following
1554 parameters.

- 1555 • *fr*: Unique identifier of the Client
- 1556 • *to*: URI of the resource targeted for the information update
- 1557 • *ri*: Identifier of the UPDATE request
- 1558 • *op*: UPDATE
- 1559 • *cn*: Information, including properties, of the resource to be updated at the target resource

1560 **8.4.2 Processing by the Server**

1561 Following the receipt of an UPDATE request, the Server may validate if the Client has the
1562 appropriate rights for updating the requested data. If the validation is successful the Server
1563 updates the target Resource information according to the information carried in *cn* parameter of
1564 the UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE
1565 request for use in the response.

1566 An UPDATE request that includes Properties that are read-only shall be rejected by the Server
1567 with an *rs* indicating a bad request.

1568 An UPDATE request shall be applied only to the Properties in the target resource visible via the
1569 applied interface that support the operation. An UPDATE of non-existent Properties is ignored.

1570 **8.4.3 UPDATE response**

1571 The UPDATE response message will include the following parameters:

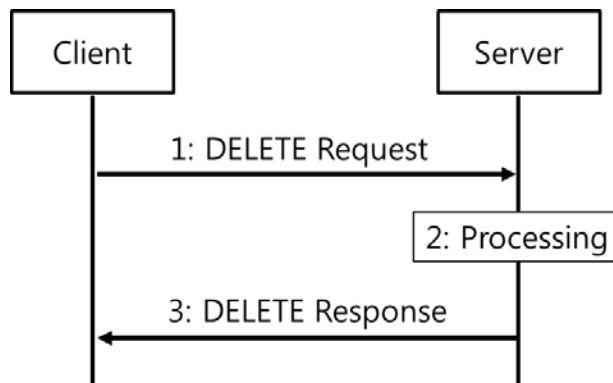
- 1572 • *fr*: Unique identifier of the Server
- 1573 • *to*: Unique identifier of the Client
- 1574 • *ri*: Identifier included in the UPDATE request
- 1575 • *rs*: The result of the UPDATE request

1576 The UPDATE response message may also include the following parameters:

- 1577 • *cn*: The Resource representation following processing of the UPDATE request

1578 **8.5 DELETE**

1579 The DELETE operation is used to request the removal of a Resource. The DELETE operation is
1580 initiated by the Client and consists of three steps, as depicted in Figure 21 and described below.



1581 **Figure 21. DELETE operation**

1583 **8.5.1 DELETE request**

1584 DELETE request message is transmitted by the Client to the Server to delete a Resource on the
1585 Server. The DELETE request message will carry the following parameters:

- 1586 • *fr*: Unique identifier of the Client
- 1587 • *to*: URI of the target resource which is the target of deletion
- 1588 • *ri*: Identifier of the DELETE request
- 1589 • *op*: DELETE

1590 **8.5.2 Processing by the Server**

1591 Following the receipt of a DELETE request, the Server may validate if the Client has the
1592 appropriate rights for deleting the identified resource, and whether the identified resource exists.
1593 If the validation is successful, the Server removes the requested resource and deletes all the
1594 associated information. The Server caches the value of *ri* parameter in the DELETE request for
1595 use in the response.

1596 **8.5.3 DELETE response**

1597 The Server shall transmit a DELETE response message in response to a DELETE request
1598 message from a Client. The DELETE response message will include the following parameters.

- 1599 • *fr*: Unique identifier of the Server
- 1600 • *to*: Unique identifier of the Client
- 1601 • *ri*: Identifier included in the DELETE request
- 1602 • *rs*: The result of the DELETE operation

1603 **8.6 NOTIFY**

1604 The NOTIFY operation is used to request asynchronous notification of state changes. Complete
1605 description of the NOTIFY operation is provided in section 11.4. The NOTIFY operation uses the
1606 NOTIFICATION response message which is defined here.

1607 **8.6.1 NOTIFICATION response**

1608 The NOTIFICATION response message is sent by an Server to notify the URLs identified by the
1609 Client of a state change. The NOTIFICATION response message carries the following parameters.

- 1610 • *fr*: Unique identifier of the Server
- 1611 • *to*: URI of the Resource target of the NOTIFICATION message
- 1612 • *ri*: Identifier included in the CREATE request
- 1613 • *op*: NOTIFY
- 1614 • *cn*: The updated state of the resource

1615 **9 Network and connectivity**

1616 **9.1 Introduction**

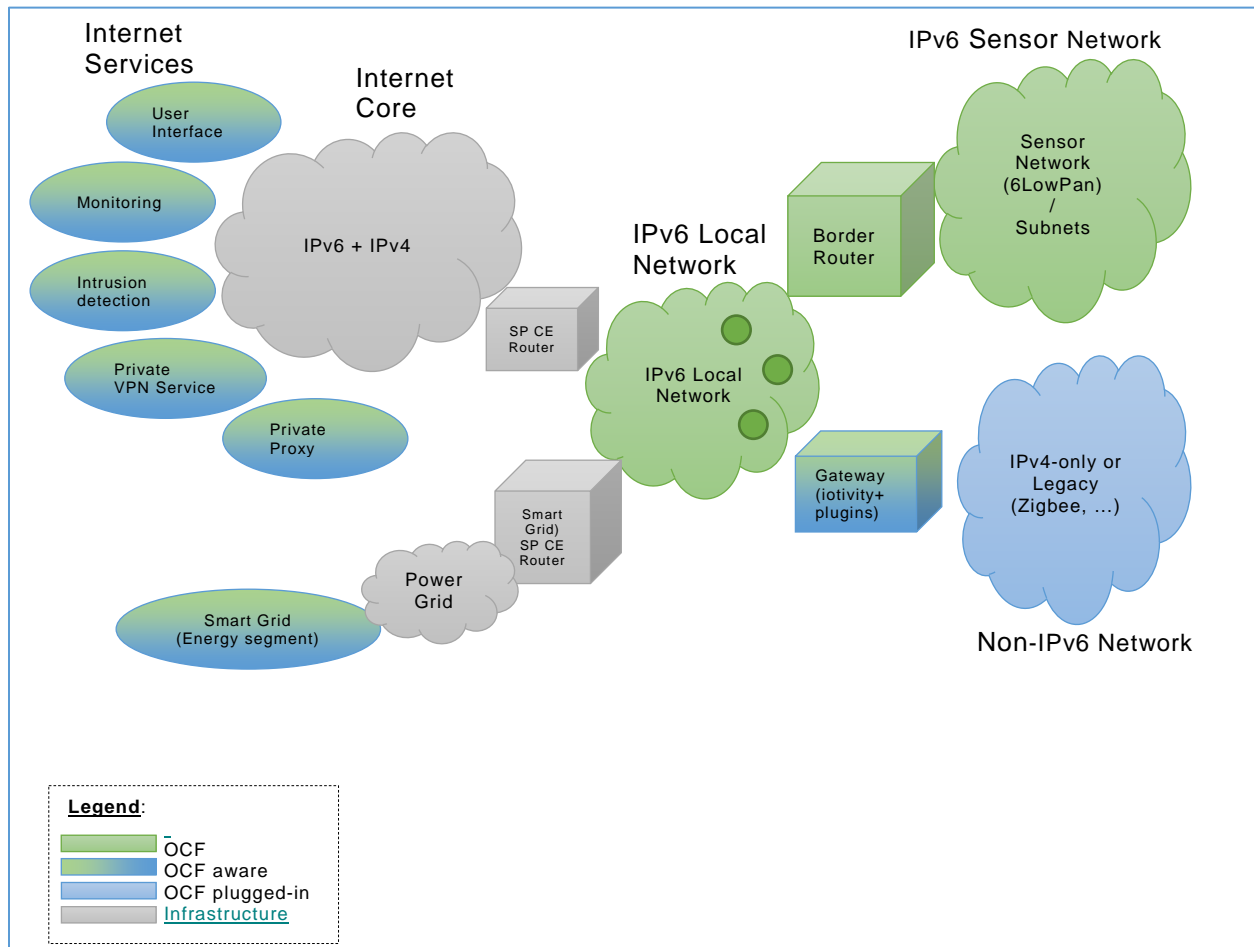
1617 The IOT environment, which the OCF is addressing, is composed of very heterogeneous systems.
1618 Because these systems are often tailored to address dedicated requirements, they are composed
1619 of very diverse products and services. Those products span from very constrained devices that
1620 run on batteries to every day high end devices available on consumer market shelves. The lack of
1621 a global standard and the need to create such a standard has led various groups to work on
1622 streamlining those technologies with well-established networking standards.

1623 The IETF recognized the market transition and realized that Ipv4 was no longer adequate. Not only
 1624 does the new scale call for a new technology, but also the manageability of even more diverse
 1625 devices, the complexity of multiple subnets and higher security and privacy needs require a whole
 1626 new set of standards. Cognizant of the existence and need for dedicated physical layer and data
 1627 link layer, the IETF set up working groups to streamline the various existing technologies at the
 1628 network layer. In accordance with these market realities, this specification also means to leverage
 1629 existing radio silicon (e.g., Bluetooth® technology, Wi-Fi, or 802.15.4) and concentrates on the
 1630 network layer and the associated data link layer adaptations produced by the IETF.

1631 **9.2 Architecture**

1632 While the aging IPv4 centric network has evolved to support complex topologies, its deployment
 1633 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More
 1634 complex network topologies, often seen in residential home, are mostly introduced through the
 1635 acquisition of additional home network devices, which rely on technologies like private Network
 1636 Address Translation (NAT). These technologies require expert assistance to set up correctly and
 1637 should be avoided in a home network as they most often result in breakage of constructs like
 1638 routing, naming and discovery services.

1639 The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices
 1640 and associated routers, but also new services introducing additional edge routers. All these new
 1641 requirements require advance architectural constructs to address complex network topologies like
 1642 the one shown in Figure 22.



1643

1644

Figure 22. High Level Network & Connectivity Architecture

1645 In terms of RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further
1646 implement various specializations of those roles. In terms of RFC 6434, IPv6 nodes assume either a
1647 router or host role. Nodes may further implement various specializations of those roles:
1648
1649 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
1650
1651 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity
1652 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).
1653 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU
1654 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),
1655
1656 • A node may translate and route messaging between IPv6 and non-IPv6 networks.
1657
1658 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
1659
1660 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity
1661 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).
1662 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU
1663 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),
1664

1665 **9.3 • A node may translate and route messaging between IPv6 and non-IPv6** 1666 **networks.IPv6 network layer requirements**

1667 **9.3.1 Introduction**

1668 Projections indicate that many 10s of billions of new IoT endpoints and related services will be
1669 brought online in the next few years. These endpoint's capabilities will span from battery powered
1670 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating
1671 over Ethernet and WiFi links.

1672 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide
1673 variety of applications such as Web browsing, email, voice, video, and critical system monitoring
1674 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which
1675 is that available address space has been consumed.

1676 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF
1677 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 1678 • Larger address space. Side-effect: greatly reduce the need for NATs.
- 1679 • More flexible addressing architecture. Multiple addresses and types per interface: Link-local,
1680 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed
1681 networks, better re-numbering capability, etc.
- 1682 • More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 1683 • Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 1684 • All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
- 1685 • Major Service Providers around the globe are deploying IPv6.

1686 **9.3.2 IPv6 node requirements**

1687 **9.3.2.1 Introduction**

1688 In order to ensure network layer services interoperability from node to node, mandating a common
1689 network layer across all nodes is vital. The protocol should enable the network to be: secure,
1690 manageable, scalable and to include constrained and self-organizing meshed nodes. OCF
1691 recommends IPv6 as the common network layer protocol to ensure interoperability across all
1692 Devices. More capable devices may also include additional protocols creating multiple-stack

1693 devices. The remainder of this section will focus on interoperability requirements for IPv6 hosts,
1694 IPv6 constrained hosts and IPv6 routers. The various protocol translation permutations included
1695 in multi-stack gateway devices may be addresses in subsequent addendums of this specification.

1696 **9.3.2.2 IP Layer**

1697 An IPv6 node should support IPv6. If a node supports IPv6, then it shall conform to the
1698 requirements for communication on the local network as follows:

- 1699 • Shall support IETF RFC 2460 “Internet Protocol version 6 Specification” and related updates
1700 as defined in section 5.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1701 • Shall support IETF RFC 4291 “IP Version 6 Addressing Architecture” and related updates as
1702 defined in section 5.9.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1703 • Shall support IETF RFC 4861 “Neighbor Discovery for IPv6” and related updates as defined in
1704 section 5.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1705 • Shall support IETF RFC 4862 “IPv6 Stateless Address Autoconfiguration” and related updates
1706 as defined in section 5.9.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1707 • Shall support IETF RFC 4443 “Internet Control Message Protocol (ICMPv6) for IPv6” [RFC4443]
1708 and related updates as defined in section 5.8 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1709 • Shall support IETF RFC 1981 “Path MTU Discovery” and related updates as defined in section
1710 5.6 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1711 • Shall support IETF RFC 4193 “Unique Local IPv6 Unicast Addresses” and related updates.
- 1712 • Shall support IETF RFC 3810 “Multicast Listener Discovery Version 2 (MLDv2) for IPv6” and
1713 related updates. In particular, shall generate new MLDv2 Report messages for every “All OCF
1714 Nodes” address FF0X::158 joined on an interface.

1715 .

1716 **9.3.3 IPv6 constrained nodes**

1717 **9.3.3.1 Requirements**

1718 An IPv6 constrained node shall support all node requirements defined in section 9.3.2. If a
1719 constrained node supports IPv6, it should use the adaptations defined as follows in order to support
1720 IPv6.

1721 **9.3.3.2 IP layer**

1722 An IPv6 constrained node should support the neighbour discovery optimization as defined in
1723 IETF RFC 6775 “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal
1724 Area Networks (6LoWPANs)”.

1725 **9.3.3.3 Sub IP layer**

- 1726 • An IPv6 constrained node on an ITU-T G.9959 network should support IETF RFC 7428 and
1727 related updates.
- 1728 • An IPv6 constrained node on an IEEE 802.15.4 network should support IETF RFC 4944 and
1729 related updates.
- 1730 • An IPv6 constrained node on a BLUETOOTH(R) Low Energy network should support
1731 IETF RFC 7668 and related updates.

1732 10 Endpoint discovery

1733 10.1 Introduction

1734 This section describes how an OCF Endpoint is discovered by another OCF Endpoint in a network.
1735 An OCF Endpoint shall support CoAP discovery.

1736 10.2 CoAP based Endpoint discovery

1737 The following describes CoAP based Endpoint discovery:

- 1738 a) Advertising or publishing Devices shall join the 'All OCF Nodes' multicast groups (as defined
1739 in [IANA IPv6 Multicast Address Space Registry]) and listen on the port 5683.
- 1740 b) Clients intending to discover resources shall join the 'All OCF Nodes' multicast groups (as
1741 defined in [IANA IPv6 Multicast Address Space Registry]).
- 1742 c) Clients shall send discovery requests (GET request) to the 'All OCF Nodes' multicast group
1743 address at port 5683. The requested URI shall be /oic/res.
- 1744 d) If the discovery request is intended for a specific resource type, the Query parameter "rt" shall
1745 be included in the request (section 6.2.1) with its value set to the desired resource type. Only
1746 Devices hosting the resource type shall respond to the discovery request.
- 1747 e) When the "rt" Query parameter is omitted, all Devices shall respond to the discovery request.
- 1748 f) Handling of multicast requests shall be as described in section 8 of IETF RFC 7252 and section
1749 4.1 in IETF RFC 6690.
- 1750 g) Devices which receive the request shall respond using CBOR payload encoding. A Device
1751 shall indicate support for CBOR payload encoding for multicast discovery as described in
1752 section 12.2.3. Later versions of the specification may support alternate payload encodings
1753 (JSON, XML/EXI, etc.).

1754

1755 Below are a few examples to search for Devices on the network:

1756 To search for all Devices on the network a Client can issue:

1757 Request

```
1758 GET /oic/res
```

1759 Response

```
1760 [  
1761 {  
1762   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",  
1763   "links": [  
1764     {  
1765       "href": "/oic/d",  
1766       "rt": ["oic.d.light", "oic.wd.d"],  
1767       "if": ["oic.if.r", "oic.if.baseline"],  
1768       "p": {"bm": 1, "sec": true, "port": 32278}  
1769     },  
1770     {  
1771       "href": "/oic/p",  
1772       "rt": ["oic.wk.p"],  
1773       "if": ["oic.if.r", "oic.if.baseline"],  
1774       "p": {"bm": 1, "sec": true, "port": 32278}  
1775     },  
1776     {  
1777       "href": "/switch",  
1778       "rt": ["oic.r.switch.binary"],  
1779       "if": ["oic.if.a", "oic.if.baseline"],  
1780       "p": {"bm": 2, "sec": true, "port": 32278}  
1781     },  
1782     {  
1783       "href": "/brightness",
```

```

1784         "rt": ["oic.r.light.brightness"],
1785         "if": ["oic.if.a", "oic.if.baseline"],
1786         "p": {"bm": 3, "sec": true, "port": 32278}
1787     }
1788 ]
1789 }
1790 ]

```

1791 To search for oic.r.switch.binary resources on the network a Client can issue:

1792 **Request**

1793 GET /oic/res?rt=oic.r.switch.binary

1794 **Response**

```

1795 [
1796 {
1797   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1798   "links": [
1799     {
1800       "href": "/switch",
1801       "rt": ["oic.r.switch.binary"],
1802       "if": ["oic.if.a", "oic.if.baseline"],
1803       "p": {"bm": 1, "sec": true, "port": 32278}
1804     }
1805   ]
1806 }
1807 ]

```

1808 Note that the examples do not indicate the multicast address and port number. The examples also do not include the
1809 accept header.

1810

1811 **11 Functional interactions**

1812 **11.1 Introduction**

1813 The functional interactions between a Client and n Server are described in section 11.2 through
1814 section 11.6 respectively. The functional interactions use CRUDN messages (section 8) and
1815 include Discovery, Notification, and Device management. These functions require support of core
1816 defined resources as defined in Table 11. More details about these resources are provided later
1817 in this section.

1818

Table 11. List of Core Resources

Pre-defined URI	Resource Name	Resource Type	Related Functional Interaction	Mandatory
/oic/res	Default	oic.wk.res	Discovery	Yes
/oic/p	Platform	oic.wk.p	Discovery	Yes
/oic/d	Device	oic.wk.d	Discovery	Yes
/oic/con	Configuration	oic.wk.con	Device Management	No
/oic/mnt	Maintenance	oic.wk.mnt	Device Management	No

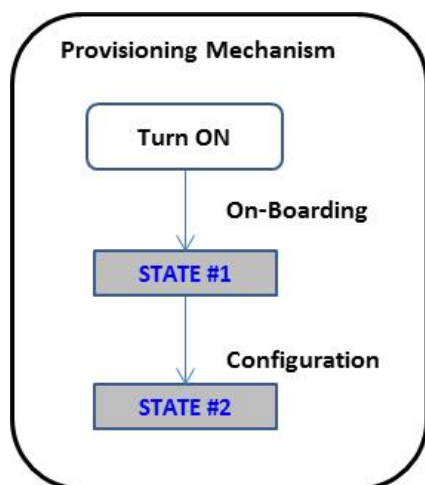
1819

1820 **11.2 Provisioning**

1821 Provisioning in Framework includes two distinct processes: onboarding and Configuration.

1822 onboarding is the process which delivers required information to a Device for joining the OCF
1823 network. When onboarding process is completed, the Device has necessary information and is
1824 able to join the OCF network (State #1 in Figure 23). Further details about provisioning can be
1825 found in OCF Security specification (Owner PSK).

1826 Configuration is the process which delivers required information to a device for accessing OCF
1827 services. At the end of the configuration process, the Device has all the necessary information and
1828 is able to access OCF services (State #2 in Figure 23).



1829

1830

Figure 23. Provisioning State Changes

1831

#1 onboarding

1832 Framework is applicable to many different types of devices with different capabilities, including
1833 devices with a rich user interface that can take inputs from the users, e.g., smartphones, as well
1834 as headless devices that have no means for receiving user inputs, e.g., sensors. Additionally, the
1835 Devices may support different communication and connectivity technologies, e.g., Bluetooth, Wi-
1836 Fi, etc. Different communication and connectivity technologies provide different onboarding
1837 mechanisms specific to that technology.

1838 Due to these differences and diversity of device capabilities, this version of specification does not
1839 mandate a particular process for onboarding, instead, specifies the state of the Device upon
1840 completion of the onboarding process.

1841 As part of the onboarding process the device acquires detailed information and required parameter
1842 values to be able to connect to the network, resulting in successful establishment of a connection
1843 to the network at the end of the onboarding process. The required information and parameters
1844 values include for example, SSID for Wi-Fi as well as authentication credentials.

1845 Later versions of this specification may specify a common process for onboarding across different
1846 communication and connectivity technologies.

1847

#2 Configuration

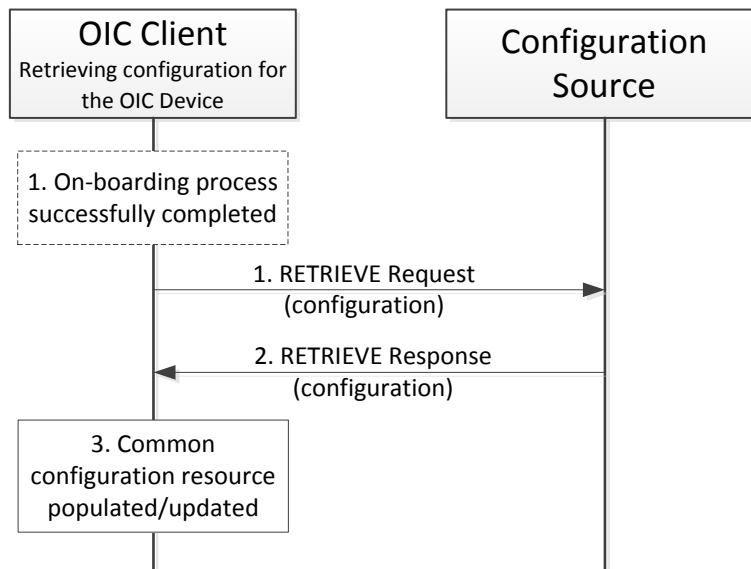
1848 Once a Device is successfully connected to the OCF network, it needs additional configuration
1849 information for accessing the OCF services or to subscribe for OCF services. The information
1850 required may include geographical location, time zone, security requirements, etc. This information
1851 may be pre-loaded on an Device, or may be acquired from a configuration service that can be
1852 located on another Device, e.g., the Configuration Source. The information regarding the
1853 configuration service resource, e.g., the URI of the Configuration Source, is pre-configured on the
1854 Device.

1855 The configuration information is also in core resource /oic/con. Upon completion of the onboarding
1856 process and as soon as the Device is connected to the network, if the configuration information is
1857 not pre-loaded, it shall initiate the configuration process, as a result of which the Device acquires
1858 the relevant configuration information, through either a pull or a push interaction, and populates
1859 its designated configuration resource with its current configured state information. The designated
1860 configuration resource maintains the latest configuration state and is the designated resource
1861 through which updates to the configuration are made.

1862 If the configuration information is not pre-loaded the Device retrieves them from the Configuration
1863 Source. During the lifetime of a Device a Client may retrieve or update the configuration state of
1864 the Device. Some of the configuration information is read only and some may be modified by
1865 Configuration Sources depending on the 'Access Modes' of properties in /oic/con resource.

1866 Figure 24 depicts the interactions triggered by a Device to retrieve its configuration information
1867 from the Configuration Source which may be located on a remote Device or locally. These
1868 interactions occur instantly following completion of onboarding process; the Device may retrieve
1869 its configuration at any time during its lifetime.

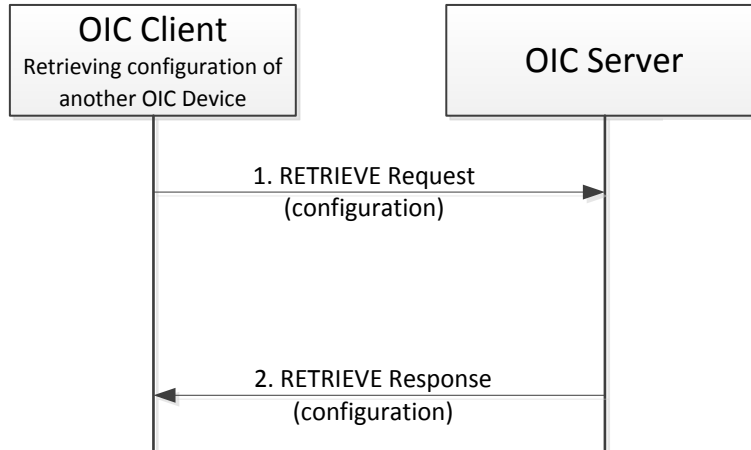
1870



1871

1872 **Figure 24. Interactions initiated by the Device to retrieve its configuration from a**
1873 **configuration source**

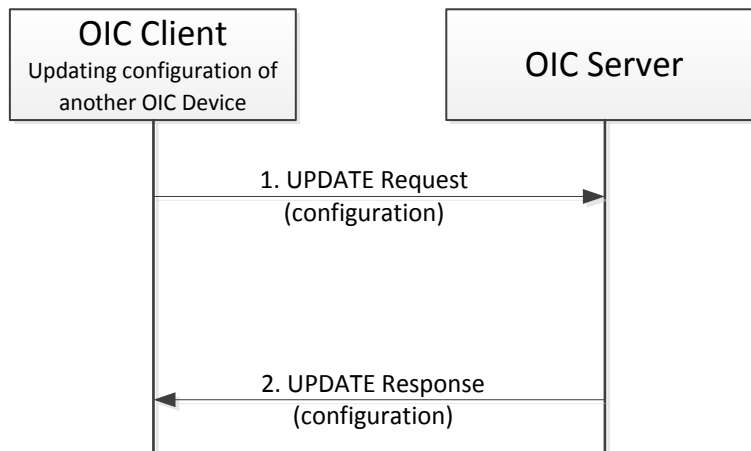
1874 Figure 25 depicts the interactions when the retrieve of configuration information is done by a Client.



1875
1876

Figure 25. Interactions for retrieving the configuration state of an Device.

1877 Figure 26 depicts the interactions when the configuration information of an Device is updated by a
1878 Client, e.g., the Configuration Source.
1879



1880
1881

Figure 26. Update of and Device configuration

1882 If Configuration is supported by a Device, i.e., the configuration information may be dynamically
1883 updated, the Core Resource /oic/con shall be supported as the designated configuration resource
1884 as described in Table 12.

1885 **Configuration Resource**

1886 A Device or a Platform may be initially configured from information that is set or provisioned at
1887 bootstrap. In addition, the Device and Platform may be configured further by an external agent
1888 post bootstrap depending on changing conditions or context. The core resource /oic/con exposes
1889 properties that may be used to effect changes in the configuration.
1890

1891 A configuration is determined by setting all the properties that collectively pertain to that
1892 configuration. The outcome of setting a new configuration is determined by the value of the specific
1893 properties in that set. Setting a new configuration through /oic/con may lead to initiation of
1894 processes that affect or create side effects in other resources.
1895

1896

Table 12. Configuration Resources

Pre-defined 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 Device is exposed. The resource properties exposed by /oic/con are listed in Table 13.	Configuration

1897

1898 Table 13 defines the oic.wk.con resource type.

1899

1900

Table 13. 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 configurable by the end user (e.g. 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 (").

1901

1902 **11.3 Resource discovery**1903 **11.3.1 Introduction**

1904 Discovery is a function which enables endpoint discovery as well as resource based discovery.
 1905 Endpoint discovery is described in detail in section 10. This section mainly describes the resource
 1906 based discovery.

1907 **11.3.2 Resource based discovery: mechanisms**1908 **11.3.2.1 Overview**

1909 As part of discovery, a Client may find appropriate information about other OCF peers. This
 1910 information could be instances of resources, resource types or any other information represented
 1911 in the resource model that an OCF peer would want another OCF peer to discover.

1912 At the minimum, Resource based discovery uses the following:

- 1913 1) A resource to enable discovery shall be defined. The representation of that resource shall
1914 contain the information that can be discovered.
- 1915 2) The resource to enable discovery shall be specified and commonly known a-priori. A Device for
1916 hosting the resource to enable discovery shall be identified.
- 1917 3) A mechanism and process to publish the information that needs to be discovered with the
1918 resource to enable discovery.
- 1919 4) A mechanism and process to access and obtain the information from the resource to enable
1920 discovery. A query may be used in the request to limit the returned information.
- 1921 5) A scope for the publication
- 1922 6) A scope for the access.
- 1923 7) A policy for visibility of the information.

1924

1925 Depending on the choice of the base aspects defined above, the Framework defines three resource
1926 based discovery mechanisms:

- 1927 • Direct discovery, where the Resources are published locally at the Device hosting the
1928 resources and are discovered through peer inquiry.
- 1929 • Indirect discovery, where Resources are published at a third party assisting with the
1930 discovery and peers publish and perform discovery against the resource to enable
1931 discovery on the assisting 3rd party.
- 1932 • Advertisement discovery, where the resource to enable discovery is hosted local to the
1933 initiator of the discovery inquiry but remote to the Devices that are publishing discovery
1934 information.

1935 A Device shall support direct discovery.

1936 **11.3.2.2 Direct discovery**

1937 In direct discovery,

- 1938 1) The Device that is providing the information shall host the resource to enable discovery.
- 1939 2) The Device publishes the information available for discovery with the local resource to
1940 enable discovery (i.e. local scope).
- 1941 3) Clients interested in discovering information about this Device shall issue RETRIEVE
1942 requests directly to the resource. The request may be made as a unicast or multicast.
1943 The request may be generic or may be qualified or limited by using appropriate queries in
1944 the request.
- 1945 4) The “server” Device that receives the request shall send a response with the discovered
1946 information directly back to the requesting “client” Device.
- 1947 5) The information that is included in the request is determined by the policies set for the
1948 resource to be discovered locally on the responding Device.

1949

1950 **11.3.2.3 Indirect discovery of Resources (resource directory based discovery)**

1951 In indirect discovery the information about the resource to be discovered is hosted on a Server
1952 that is not hosting the resource. See section 11.3.6 for details on resource directory based
1953 discovery.

1954 In indirect discovery:

- 1955 a) The resource to be discovered is hosted on a Device that is neither the client initiating
1956 the discovery nor the Device that is providing or publishing the information to be
1957 discovered. This Device may use the same resource to provide discovery for multiple
1958 agents looking to discover and for multiple agents with information to be discovered.
- 1959 b) The Device to be discovered or with information to discover, publishes that information
1960 with resource to be discovered on a different Device. The policies on the information
1961 shared including the lifetime/validity are specified by the publishing Device. The
1962 publishing Device may modify these policies as required.
- 1963 c) The client doing the discovery may send a unicast discovery request to the Device
1964 hosting the discovery information or send a multicast request that shall be monitored and
1965 responded to by the Device. In both cases, the Device hosting the discovery information
1966 is acting on behalf of the publishing Device.
- 1967 d) The discovery policies may be set by the Device hosting the discovery information or by
1968 the party that is publishing the information to be discovered. The discovery information
1969 that is returned in the discovery response shall adhere to the policies that are in effect at
1970 the time of the request.
- 1971

1972 **11.3.2.4 Advertisement Discovery**

1973 In advertisement discovery:

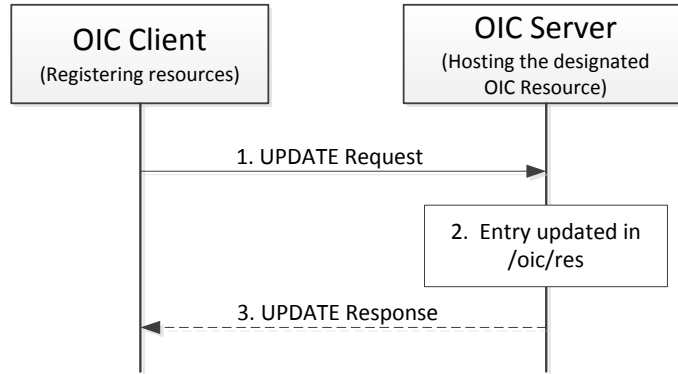
- 1974 a) The resource to enable discovery is hosted local to the Device that is initiating the discovery
1975 request (client). The resource to enable discovery may be an Core Resource or discovered
1976 as part of a bootstrap.
- 1977 b) The request could be an implementation dependent lookup or be a local RETRIEVE request
1978 against the resource that enables discovery.
- 1979 c) The Device with information to be discovered shall publish the appropriate information to
1980 the resource that enables discovery.
- 1981 d) The publishing Device is responsible for the published information. The publishing Device
1982 may UPDATE the information at the resource to enable discovery based on its needs by
1983 sending additional publication requests. The policies on the information that is discovered
1984 including lifetime is determined by the publishing Device.

1985

1986 **11.3.3 Resource based discovery: Information publication process**

1987 The mechanism to publish information with the resource to enable discovery can be done either
1988 locally or remotely. The publication process is depicted in Figure 27. The Device which has
1989 discovery information to publish shall a) either update the resource that enables discovery if
1990 hosted locally or b) issue an UPDATE request with the information to the Device which hosts the
1991 resource that enables discovery. The Device hosting the resource to enable discovery
1992 adds/updates the resource to enable discovery with the provided information and then responds
1993 to the Device which has requested the publication of the resource with an UPDATE response.

1994

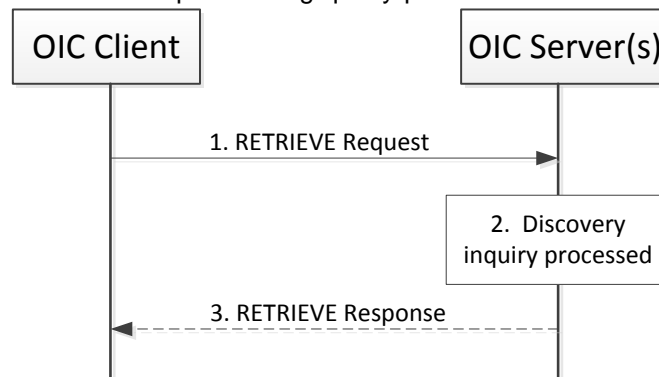


1995
1996

Figure 27. Resource based discovery: Information publication process

1997 **11.3.4 Resource based discovery: Finding information**

1998 The discovery process (Figure 28) is initiated as a RETRIEVE request to the resource to enable
 1999 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as
 2000 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the
 2001 support in the data connectivity layer. The response to the request has the information to be
 2002 discovered based on the policies for that information. The policies can determine which information
 2003 is shared, when and to which requesting agent. The information that can be discovered can be
 2004 resources, types, configuration and many other standards or custom aspects depending on the
 2005 request to appropriate resource and the form of request. Optionally the requester may narrow the
 2006 information to be returned in the request using query parameters in the URI query.



2007
2008
2009

Figure 28. Resource based discovery: Finding information

2010 **Discovery Resources**

2011 Some of the Core Resources shall be implemented on all Devices to support discovery. The Core
 2012 Resources that shall be implemented to support discovery are:

- 2013 ● /oic/res for discovery of resources
- 2014 ● /oic/p for discovery of platform
- 2015 ● /oic/d for discovery of device information

2016 Details for these mandatory Core Resources are described in Table 14

2017 Platform resource –

2018 The OCF recognizes that more than one instance of Device may be hosted on a single platform.
 2019 Clients need a way to discover and access the information on the platform. The core resource,
 2020 /oic/p exposes platform specific properties. All instances of Device on the same Platform shall
 2021 have the same values of any properties exposed (i.e. an Device may choose to expose optional
 2022 properties within /oic/p but when exposed the value of that property should be the same as the
 2023 value of that property on all other Devices on that Platform)

2024
 2025 **Device resource**

2026 The device resource shall have the pre-defined URI /oic/d. The resource /oic/d exposes the
 2027 properties pertaining to a Device as defined in Table 14. The properties exposed are determined
 2028 by the specific instance of Device and defined by the resource type(s) of /oic/d on that Device.
 2029 Since all the resource types of /oic/d are not known a priori, the resource type(s) of /oic/d shall be
 2030 determined by discovery through the core resource /oic/res. The device resource /oic/d shall have
 2031 a default resource type that helps in bootstrapping the interactions with this device (the default
 2032 type is described in Table 14.)

2033
 2034 **Protocol indication**

2035 A Device may need to support different messaging protocols depending on requirements for
 2036 different application profiles. For example, the Smart Home profile may use CoAP and the
 2037 Industrial profile may use DDS. To enable interoperability, a Device uses the protocol indication
 2038 to indicate the transport protocols they support and can communicate over.

2039

2040

Table 14. Mandatory discovery Core Resources

Pre-define d 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 Server is discovered and introspected for available resources. /oic/res shall expose the resources that are discoverable on a Device. When an Server receives a RETRIEVE request targeting /oic/res (e.g., GET /oic/res), it shall respond with the link list of all the discoverable resources of itself. The /oic/d and /oic/p are discoverable resources, hence their links are included in /oic/res response. The resource properties exposed by /oic/res are listed in Table 15.	Discovery
/oic/p	Platform	oic.wk.p	oic.if.r	The discoverable resource through which platform specific information is discovered. The resource properties exposed by /oic/p are listed in Table 17	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 Device instance. The resource properties exposed by /oic/d are listed in Table 17 /oic/d may have one or more resource types that are specific to 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 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

2041

2042 Table 15 defines oic.wk.res resource type.

2043

Table 15. oic.wk.res resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R	no	Human-friendly name defined by the vendor
Device Identifier	di	UUID			R	yes	The device identifier as indicated by the /oic/d resource of the Device. There may be multiple "di" instances in /oic/res but each "di" shall have a unique value. This "di" value uniqueness implies that the resources of a device shall be grouped together under a single "di".
Links	links	array	See 7.8.2		R	yes	The array of Links describes the URI, supported resource types and interfaces, and access policy.
Messaging Protocol	mpro	SSV			R	No	String with Space Separated Values (SSV) of messaging protocols supported as a SI Number from Table 16 For example, "1 and 3" indicates that the Device supports coap and http as messaging protocols.

2044 A Device which wants to indicate its messaging protocol capabilities may add the property 'mpro'
 2045 in response to a request on /oic/res. A Device shall support CoAP based discovery as the baseline
 2046 discovery mechanism (see section 10.2). A Client which sees this property in a discovery response
 2047 can choose any of the supported messaging protocols for communicating with the Server for further
 2048 messages. For example, if a Device supporting multiple protocols indicates it supports a value of
 2049 "1 3" for the 'mpro' property in the discovery response, then it cannot be assumed that there is an
 2050 implied ordering or priority. But a vertical service specification may choose to specify an implied
 2051 ordering or priority. If the 'mpro' property is not present in the response, A Client shall use the
 2052 default messaging protocol as specified in the vertical specification for further communication.
 2053 Table 16 provides an OCF registry for protocol schemes.

2054

Table 16. Protocol scheme registry

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

2055 Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by a Client to form
 2056 requests in a different messaging protocol other than discovery is out of scope.

2057

2058 The following applies to the use of /oic/d as defined above:

- 2059 • A vertical may choose to expose its Device Type (e.g., refrigerator or A/C) by adding the Device
2060 Type to the list of Resource Types associated with /oic/d.
 - 2061 ○ For example; rt of /oic/d becomes ["oic.wk.d", "oic.d.<thing>"]; where “oic.d.<thing>”
2062 is defined in another spec such as the Smart Home vertical.
 - 2063 ○ This implies that the properties exposed by /oic/d are by default the mandatory
2064 properties in Table 17.
- 2065 • A vertical may choose to extend the list of properties defined by the Resource Type 'oic.wk.d'.
2066 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The
2067 mandatory properties defined in Table 17 shall always be present.

2068 Note:

2069 As per existing Core specification definitions the resource type ID may be a list of resource type IDs; when that is the
2070 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.
2071 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
2072 for the rt listed in the query.

2073 Table 17 oic.wk.d resource type definition defines the base resource type for the /oic/d resource.
2074

2075

Table 17. oic.wk.d resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
(Device) Name	n	string			R	no	Human friendly name defined by the vendor."
Spec Version	icv	string			R	yes	Spec version of the core specification this device is implemented to, The syntax is "core.<major>.<minor>.<sub-version>" where <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. This version of the specification the string value shall be "core.1.1.0".
Device ID	di	UUID			R	yes	Unique identifier for Device. This value shall be as defined in [OCF Security] for DeviceID.
Data Model Version	dmv	CSV			R	yes	Spec version of the Resource Specification to which this device data model is implemented; if implemented against a Vertical specific resource specification, then the Spec version of the vertical specification this device model is implemented to. The syntax is a comma separated list of " <res>.<major>.<minor>.<sub-version> or <vertical>.<major>.<minor>.<sub-version>. <res> is the string "res" and <vertical> is the name of the vertical defined in the Vertical specific resource specification. The <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. This

							version of the specification the string value shall be "res.1.1.0".
--	--	--	--	--	--	--	---

2076

2077 The additional resource type(s) of the /oic/d resource are defined by the vertical specification.

2078

2079 Table 18 defines oic.wk.p resource type.

2080

2081

Table 18. oic.wk.p resource type definition

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Platform ID	pi	string			R	yes	Unique identifier for the physical platform (UUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC.
Manufacturer Name	mnmn	string			R	yes	Name of manufacturer
Manufacturer Details Link	mnml	URI			R	no	Reference to manufacturer, represented as a URI
Model Number	mnmo	string			R	no	Model number as designated by manufacturer
Date of Manufacture	mnmt	date		Time <i>(show RFC)</i>	R	no	Manufacturing date of device
Platform Version	mnpv	string			R	no	Version of platform – string (defined by manufacturer)
OS Version	mnos	string			R	no	Version of platform resident OS – string (defined by manufacturer)
Hardware Version	mnhw	string			R	no	Version of platform hardware
Firmware version	mnfv	string			R	no	Version of device firmware
Support link	mnsi	URI			R	no	URI that points to support information from manufacturer
SystemTime	st	datetime			R	no	Reference time for the device. The format is restricted to the 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.
Vendor ID	vid	string			R	no	Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it.

2082

2083 **Composite Device**

2084 A physical device may be modelled as a single device or as a composition of other devices. For
 2085 example a refrigerator may be modelled as a composition, as such part of its definition of may
 2086 include a sub-tending thermostat device which itself may be composed of a sub-tending
 2087 thermometer device.

2088 There may be more than one way to model an server as a composition. One example method
 2089 would be to have Platform which represents the composite device to have more than one instance
 2090 of a Device on the Platform. Each Device instance represents one of the distinct devices in the
 2091 composition. Each instance of Device may itself have or host multiple instances of other resources.

2092 An implementation irrespective of how it is composed shall only expose a single instance of /oic/d
 2093 with an 'rt' of choice for each logical Server.

2094 Thus, for the above refrigerator example if modeled as a single Server; /oic/res would expose
 2095 /oic/d with a resource type name appropriate to a refrigerator. The sub-tending thermostat and
 2096 thermometer devices would be exposed simply as instances of a resource with a device
 2097 appropriate resource type with an associated URI assigned by the implementation; e.g.,
 2098 /MyHost/MyRefrigerator/Thermostat and /MyHost/MyRefrigerator/Thermostat/Thermometer.

2099

2100 **11.3.5 Resource discovery using /oic/res**

2101 Discovery using /oic/res is the default discovery mechanism that shall be supported by all Devices
 2102 as follows:

- 2103 a) Every Device updates its local /oic/res with the resources that are discoverable (see section
 2104 7.3.2.2). Every time a new resource is instantiated on the Device and if that resource is
 2105 discoverable by a remote Device then that resource is published with the /oic/res resource that
 2106 is local to the Device (as the instantiated resource).
- 2107 b) An Device wanting to discover resources or resource types on one or more remote Devices
 2108 makes a RETRIEVE request to the /oic/res on the remote Devices. This request may be sent
 2109 multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request
 2110 may optionally be restricted using appropriate clauses in the query portion of the request.
 2111 Queries may select based on resource types, interfaces, or properties.
- 2112 c) Query applies to the representation of the resources. /oic/res is the only resource whose
 2113 representation has "rt". So /oic/res is the only resource that can be used for Multicast discovery
 2114 at the transport protocol layer.
- 2115 d) The Device receiving the RETRIEVE request responds with a list of resources, the resource
 2116 type of each of the resources and the interfaces that each resource supports. Additionally

2117 information on the policies active on the resource can also be sent. The policy supported
2118 includes observability and discoverability. (More details below)

2119 e) The receiving Device may do a deeper discovery based on the resources returned in the
2120 request to /oic/res.

2121

2122 The information that is returned on discovery against /oic/res is at the minimum:

- 2123 • The URI (relative or fully qualified URL) of the resource
- 2124 • The Resource Type of each resource. More than one Resource Type may be returned if the
2125 resource enables more than one type. To access resources of multiple types, the specific
2126 resource type that is targeted shall be specified in the request.
- 2127 • The Interfaces supported by that Resource. Multiple interfaces may be returned. To access a
2128 specific interface that interface shall be specified in the request. If the interface is not specified,
2129 then the Default Interface is assumed.
- 2130 • Policies defined against that resource. These policies may be security related, access modes,
2131 types of interactions, etc. In addition to the request/response type of interaction, the
2132 specification allows the resource to be “observed” (section 11.4.2).

2133

2134 The JSON schemas for discovery using /oic/res are described in D.8. Also refer to Section 10
2135 (Endpoint Discovery) for details of Multicast discovery using /oic/res on a CoAP transport.

2136 After performing discovery using /oic/res, Clients may discover additional details about Server by
2137 performing discovery using /oic/p, /oic/rts etc. If a Client already knows about Server it may
2138 discover using other resources without going through the discovery of /oic/res.

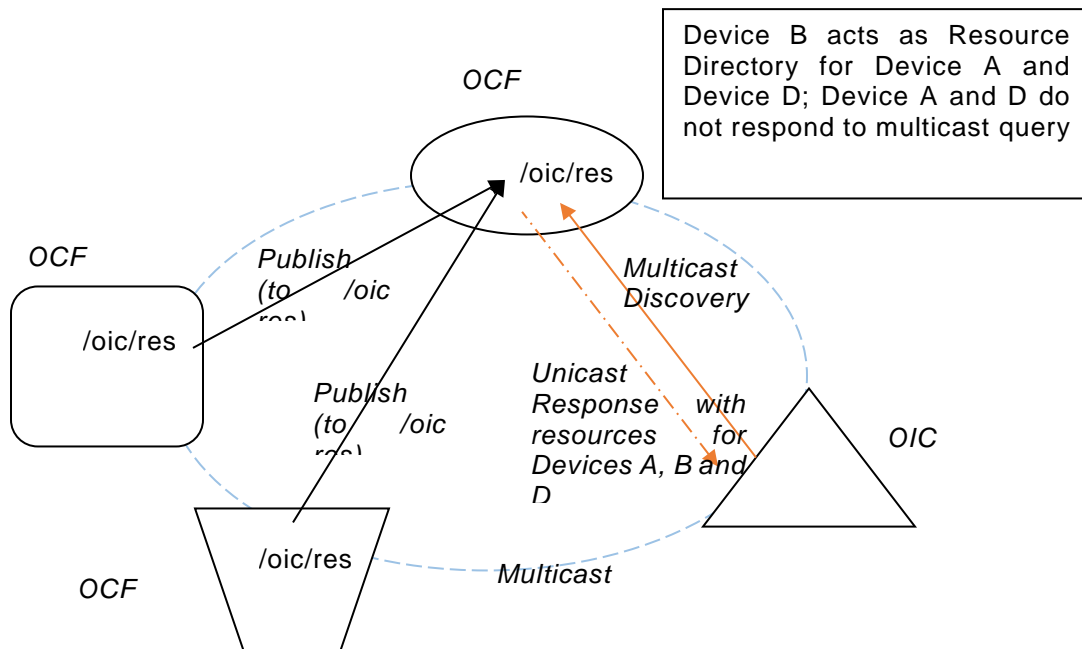
2139 **11.3.6 Resource directory (RD) based discovery**

2140 **11.3.6.1 Introduction**

2141 **11.3.6.1.1 Indirect discovery for lookup of the resources**

2142 Direct discovery is the mechanism used currently to find resources in the network. When needed,
2143 resources are queried at a particular node directly or a multicast packet is sent to all nodes. Each
2144 queried node responds directly with its discoverable resources to the discovering device.
2145 Resources available locally are registered on the same device.

2146 In some situations, one of the other mechanisms described in section 11.3.2.3, called indirect
2147 discovery, may be required. Indirect discovery is when a 3rd party device, other than the
2148 discovering device and the discovered device, assists with the discovery process. The 3rd party
2149 only provides information on resources on behalf of another device but does not host resources
2150 on part of that device.



2151

2152

Figure 29. Indirect discovery of resource by resource directory

2153 Indirect discovery is useful for a resource constrained device that needs to sleep to manage power
 2154 and cannot process every discovery request, or when devices may not be on the same network
 2155 and requires optimization for discovery. Once resources are discovered using indirect discovery
 2156 then the access to the resource is done by a request directly to the Device that hosts that resource.

2157 11.3.6.1.2 Resource directory

2158 A resource directory (RD) is an Device that assists with indirect discovery. A RD can be queried
 2159 at its /oic/res resource to find resources hosted on other Devices. These Devices can be sleepy
 2160 nodes or any other device that cannot or may not respond to discovery requests. Device can
 2161 publish all or partial list of resources they host to a RD. The RD then responds to queries for
 2162 Resource discovery on behalf of the publishing Device (for example: when a Device may go to
 2163 sleep). For general Resource discovery, the RD behaves like any other Server in responding to
 2164 requests to /oic/res.

2165 Any Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The Devices
 2166 that want to discover RDs shall use this resource and one of the Resource discovery mechanisms
 2167 to discover the RD and get the parameters of the RD. The information discovered through this
 2168 resource shall be used to select the appropriate RD to use for resource publication. The bias
 2169 information shall include the following criteria: power source (AC, battery powered or safe/reliable),
 2170 connectivity (wireless, wired), CPU, memory, load statistics (processing publishing and query from
 2171 the devices). In addition, the RD shall return a bias factor that ranges from 0 to 100. Optionally,
 2172 the RD may also return a context - the value which shall be a string and semantics of the context
 2173 are not discussed in this document but it is expected that the context will be used to establish a
 2174 domain, region or some such scope that is meaningful to the application, deployment or usage.

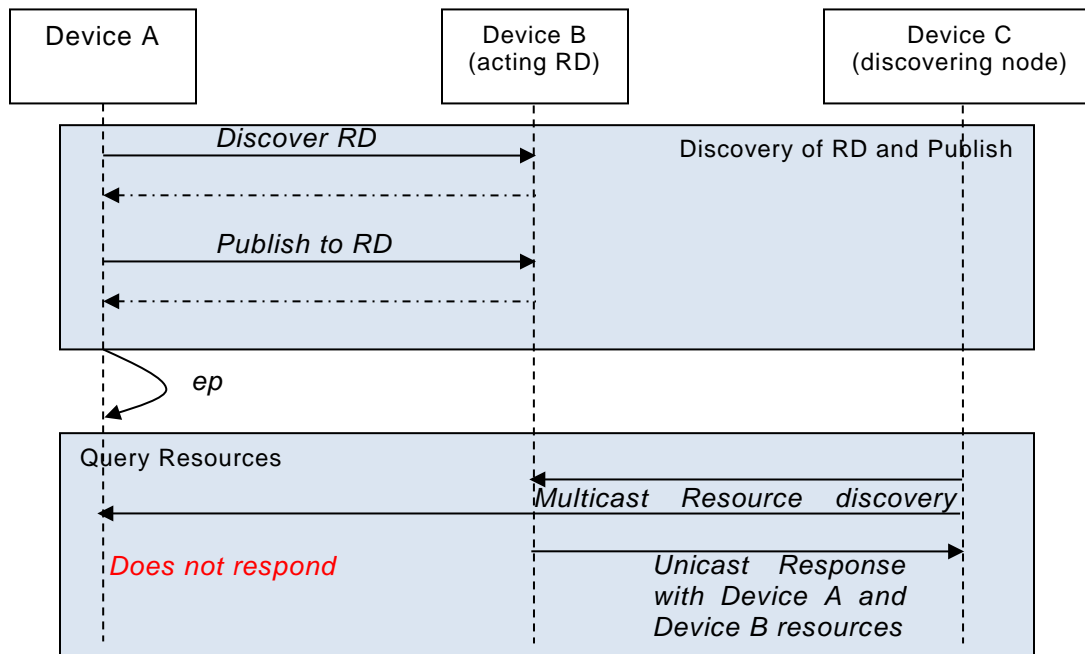
2175 Using these criteria or the bias factor, the Device shall select one RD (per context) to publish its
 2176 resources. A context describes the state of an OCF Device with respect to Resource discovery. A
 2177 context is usually determined at deployment and from application requirements. An example of a
 2178 context could be a multicast group- a Device that is a member of more than one multicast group
 2179 may have to find and select a RD in each of the multicast groups (i.e. per context) to publish its
 2180 information. The Device may choose other RDs during its lifetime but a Device shall not publish

2181 its resource information to more than one RD Devices such as TV, network router, desktop will
2182 have higher weightage or bias factor compared to mobile phone device.

2183 **11.3.6.2 The remainder of this section is divided into two parts. The first part covers**
2184 **discovering of the RD and publishing, updating and deleting of resources for**
2185 **the constrained/sleepy device. The second part covers the replies of the RD to**
2186 **queries from devices with the aim to discover resources. Resource directory**
2187 **discovery**

2188 **11.3.6.2.1 Discovering a resource directory**

2189 A RD in the OCF network shall support RD discovery, shall provide the facility to allow devices to
2190 publish their resource information to a RD, to update resource information in a RD and to delete
2191 resource information from a RD.



2192

2193

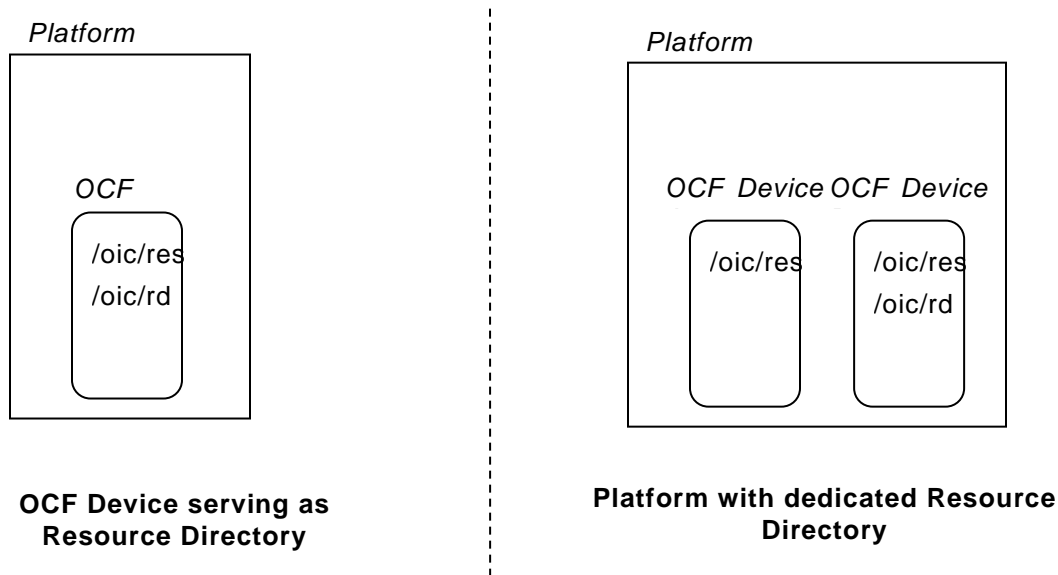
Figure 30. RD discovery and RD supported query of resources support

2194 As shown in Figure 30, the Device that wishes to advertise its resources: first discovers a resource
2195 directory and then publishes the desired resource information. Once a set of resources have been
2196 published to a RD then the publishing device shall not respond to multicast Resource discovery
2197 queries for those published resources when the RD is on the same multicast domain. In that case,
2198 only the RD shall respond to multicast Resource discovery requests on the resource published to
2199 it.

2200 An OCF network allows for more than one device acting as a RD. The reason to have multiple RD
2201 support is to make network scalable, handle network failures and centralized device failure
2202 bottleneck. This does not preclude a scenario where a use case or deployment environment may
2203 require single device in the environment to be deployed as the only resource directory (e.g.
2204 gateway model). There may be more than one Device acting as RD on a Platform.

2205 Discovering of an RD may result in responses from more than one RD. The discovering device
2206 shall select a RD. The selection may be based on the weightage parameter(s) provided in the
2207 response from the RD.

2208 An RD will be application agnostic i.e., application should not be aware whether resource directory
 2209 was queried to get the resource information. All the handling of the retrieval is kept opaque to the
 2210 application. A Client that performs Resource discovery uses an RD just like it may use any other
 2211 Server for discovery. It may send a unicast request to the RD when it needs only the resource
 2212 advertised on the RD or do a multicast query when it does not require or have explicit knowledge
 2213 of an RD.



2214
 2215 **Figure 31. Resource Direction Deployment Scenarios**

2216 Resource directory can also be discovered in the following manners:

- 2217 • Pre-configuration: Devices wishing to publish resource information may be configured a priori
 2218 with the information (e.g. IP address, port, transport etc.) of a specific resource directory. This
 2219 pre-configuration may be done at onboarding or may be updated on the device using an out-
 2220 of-band method. This pre-configuration may be done by the manufacturer or by the user/device
 2221 manager.
- 2222 • Query-oriented: A Client wanting to discover resource directories using query-oriented
 2223 discovery (i.e. pull) shall issue multicast Resource discovery request directed to the /oic/rd
 2224 resource. Only the devices that hosts a /oic/rd resource shall respond to this query. The
 2225 response shall include information about the RD (as defined by the resource type) and
 2226 weightage parameters to allow the discovering device to select between RDs (see details in
 2227 RD selection section). The /oid/rd resource shall be instantiated on the OCF Devices acting as
 2228 a resource directory. The /oic/rd schema is as defined in D.12.
- 2229 • Advertisement: An RD may advertise about itself to devices. It is an advertisement packet. The
 2230 devices that are already publishing to a RD may use this as a heartbeat message of the RD. If
 2231 the RD advertisement does not arrive at a stipulated interval, publishing device starts searching
 2232 for other RDs in the network, as this is a signal that RD is not online. Other usage of this
 2233 message is it serves as an advertisement for a device seeking a RD to publish their resources.
 2234 The details from the advertisement can then be used to query directly to a RD to get weightage
 2235 details instead of sending a multicast packet in a network. As it is intended this is sent at a
 2236 regular interval and does not include weightage information to keep packet sizes small.
- 2237 • One of the important benefits of an RD is to make services discoverable in networks that don't
 2238 support site wide multicast but do support site wide routing. An example of such a network is
 2239 Homenet .To enable an RD function across such a network a site discovery mechanism is
 2240 needed to discover the RD service (IP address & port number). Homenets that support hybrid

2241 proxy (IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00) allow site wide discovery based on
 2242 dns-sd/mDNS. In order to make itself discoverable beyond the link local scope, an RD with a
 2243 routable ip address shall implement the mDNS responder requirements defined in
 2244 IETF RFC 6762. The RD shall respond to mDNS queries of type PTR and with a service name
 2245 equal to "_rd._sub._oic._udp.local". The response shall include all routable IP addresses.
 2246 Devices with a routable ip address shall discover all available RD instances by issuing a DNS-
 2247 SD's PTR lookup as defined in IETF RFC 6763 with as service name service name
 2248 "_rd._sub._oic._udp.local". The response shall include all routable addresses/port pair through
 2249 which the RD service is made accessible.

2250 **11.3.6.2.2 Resource directory selection process**

2251 **11.3.6.2.2.1 Selection criteria**

2252 When a device discovers more than one RD then it shall decide to use one of these RDs based on
 2253 the selection criteria described here. A device shall use or publish information to only one RD
 2254 within a multicast domain at a given time. This is to minimize the burden of processing duplicate
 2255 information in the Resource discovery phase.

2256 There two ways to select an RD. One is based on a bias factor (RD generated) and the other is
 2257 based on clients determination based on granular parameters provided by the server (client/device
 2258 generated). Devices may use one or both methods to select an RD.

2259 *Bias factor:* The bias factor is a server generated positive number in the range of 0 to 100, where
 2260 0 is the lowest to 100 being the highest. If two RDs have the same bias factor then the selecting
 2261 device may choose either based auxiliary criteria or at random. Either way only one RD shall be
 2262 selected and used at a time. No specific method is defined in this specification to determine the
 2263 bias factor for an RD. The number may be a pre-configured value at the time of onboarding or
 2264 subsequent configuration of the RD or may be based on a formula determined by the
 2265 implementation of the RD. (OCF will provide a standard formula for this calculation in a future
 2266 version or release of specification).

2267 The bias factor shall be calculated by the RD by adding the contribution values determined for
 2268 each of the parameters in Table 19 and divided by the number of parameters. An RD may advertise
 2269 a bias factor larger than the calculated value when there is reason to believe that the RD is highly
 2270 capable for example an installed service provider gateway.

2271 *Parameters:* Optionally, parameters defined in Table 19 (like direct power supply, network
 2272 connectivity, load conditions, CPU power, memory, etc.) may be returned in the discovery
 2273 response. Discovering device may use the details to make granular selection decisions based on
 2274 client defined policies and criteria that use the RD parameters. For example, a device in an
 2275 industrial deployment may not weight power connectivity high but another in home environments
 2276 may give more weightage for power.

2277 **Table 19: 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 = $100 - ((L1*3 + L5*7 + L15*10)/3)$

2278

2279 **11.3.6.2.2.2 Selection scenarios**

2280 The device that wants to use an RD will use the endpoint discovery to find zero or more RDs on
2281 the network. After discovering the RDs, the device needs to select an RD of all found RDs on the
2282 network. The selection based on the bias factor will ensure that an Device can judge if the found
2283 RD is suitable for its needs.

2284 The following situation can occur during the selection of an RD:

- 2285 1) A single or multiple RDs are present in the network
- 2286 2) No RD is present in the network
- 2287 3) an additional RD arrives on the network

2288

2289 In the first scenario the RDs are already present. If a single RD is detected then that RD can be
2290 used . When multiple RDs are detected the Device uses the bias information to select the RD.

2291

2292 In the second scenario, device will listen to the advertisement of the devices that hosts the RDs.
2293 Once an RD advertisement packet is received it judges if the bias criteria are met and starts using
2294 the RDs.

2295

2296 In the third scenario the Device has already published its resources to an existing RD. In this
2297 scenario it discovers a new RD on the network.

2298 After judging the bias factor the Device may choose to move to the new RD.

2299

2300 **11.3.6.3 If the decision is made to select the new RD, the then Device shall delete its**
2301 **resource information from the current used RD and then after removal publish**
2302 **the information to the new RD. During the transition period the Device itself**
2303 **shall respond to Resource discovery requests. Resource publishing**

2304 **11.3.6.3.1 Publish resources**

2305 **11.3.6.3.1.1 Overview**

2306 After the selection process of a RD, a device may choose one of the following mechanisms:

- 2307 • Push its resources information to the selected RD or
- 2308 • Request the RD to pull the resource information by doing a unicast discovery request against
2309 its /oic/res

2310 The publishing device may decide to publish all resources or few resources on the resource
2311 directory. The publishing device shall only publish resources that are otherwise published to its
2312 own /oic/res. A publishing device may respond to discovery requests (on its /oic/res resource) for
2313 the resources it does not publish to a RD. Nonetheless, it is highly recommended that when an RD
2314 is used, all discoverable resources on the publisher be published to the RD.

2315 **11.3.6.3.1.2 Publish: Push resource information**

2316 Resource information is published using an UPDATE CRUDN operation to /oic/rd using the
2317 resource type oic.wk.rdpub and the oic.if.baseline interface.

2318 Once a publishing device has published resources to a RD, it may not respond to the multicast
2319 discovery queries for the same resources against its own /oic/res, especially when on the same
2320 multicast domain as the RD. After publishing resources, it is a RD responsibility to reply to the
2321 queries for the published resources.

2322 If the publishing device is in sleep mode and a RD has replied on behalf of the publishing device,
2323 then a discovering device will try to access resource on the provided URI.

2324 There is another possibility that the resource directory and the publishing device both respond to
2325 the multicast query from the discovering device. This will create a duplication of the packet but is
2326 an alternate that may be used for non-robust network. It is not a recommended option but for
2327 industrial scenarios, this is one of the possibilities. Either way, discovering clients shall always be
2328 prepared to process duplicate information in responses to multicast discovery request. The /oic/rd
2329 schema is as defined in D.12 to specify publishing (oic.rd.publish) to the /oic/rd resource.

2330 **11.3.6.3.2 Update resource information**

2331 Server will hold the publish resource information till the time specified in the ttl field. A device can
2332 send update if it seeks a RD to keep holding resources and reply to queries on its behalf. Update
2333 can be used for updating about all resources that are published on a RD or can use to do per
2334 resource published.

2335 Updates are done using the same resource type and interface as for the initial publish but only the
2336 information to be updated is provided in the payload.

2337 **11.3.6.3.3 Delete resource information**

2338 A resource information hold at the resource directory can be removed anytime by the publishing
2339 device. It can be either for the whole device information or for a particular resource. This resource
2340 should be only allowed when device meets a certain requirement, as it can create potential security
2341 issue.

2342 The delete is done using the device ID “id” as the tag in DELETE request query when all the
2343 resource information from the device is to be deleted. In the case of a specific resource then the
2344 DELETE request shall include the instance “ins” tag along with the device ID in the query.

2345 Selective deletion of information for individual resources is not possible the case where the RD
2346 pull the resource information. The publishing device can request a delete but only for all the
2347 resource information that the RD has pulled from that device. In this case, the DELETE request
2348 has the device ID “id” tag in the query.

2349 **11.3.6.3.4 Transfer resource information from one RD to another**

2350 When a publishing device identifies an RD that is better suited, it may decide to publish to that RD.
2351 Since the device shall publish to only one RD at a time, the client shall ensure that previously
2352 published information is deleted from the currently used RD before publishing to the newly selected
2353 RD. The deletion of the resource may be done either by allowing the TTL to expire or explicitly
2354 deleting the resource information.

2355 RDs shall not communicate resource information between themselves. It is the client’s
2356 responsibility to choose the RD and to manage the published resources.

2357 **11.3.6.4 Resource discovery**

2358 **11.3.6.4.1 Query and retrieving of the resources**

2359 The query based discovery process remains the same as that in the absence of an RD. Resources
2360 may be discovered by querying the /oic/res resource by sending a multicast or unicast request. In
2361 the case of a multicast discovery request, an RD will respond for the device that hosts the
2362 resources. Clients shall be prepared to process duplicate resource information from more than one
2363 RD responding with the same information or from an RD and the hosting device (publishing the
2364 resource information) both responding to the request. Interaction with resources discovered using
2365 the RD is done using the same mechanism and methods as with resources discovered by querying
2366 the /oic/res resource of the device hosting the resources (e.g., connect to the resource and perform
2367 CRUDN operations on the resource).

2368 **11.4 Notification**

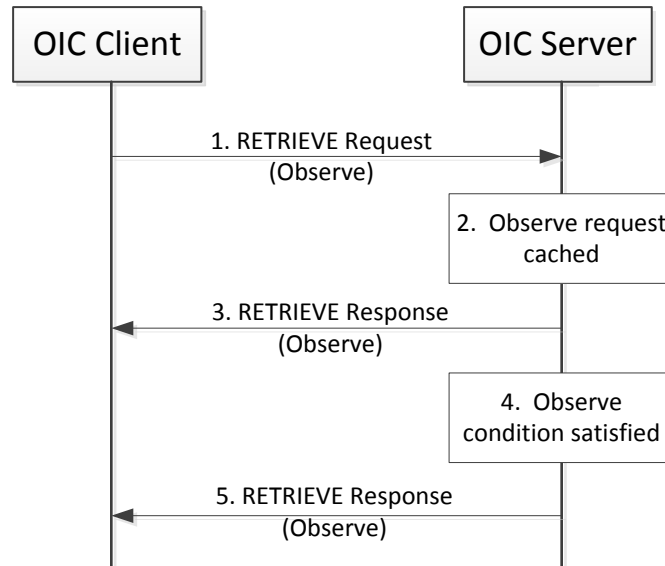
2369 **11.4.1 Overview**

2370 An Server shall support NOTIFY operation to enable a Client to request and be notified of desired
2371 states of one or more Resources in an asynchronous manner. Section 11.4.2 specifies the observe
2372 mechanism in which updates are delivered to the requester.

2373 **11.4.2 Observe**

2374 In observe mechanism the Client utilizes the RETRIEVE operation to require the Server for updates
2375 in case of Resource state changes. The Observe mechanism consists of five steps which are
2376 depicted in Figure 32 and described below.

2377 Note: the observe mechanism can only be used for a resource with a property of observable
2378 (section 7.3.2.2).



2379

2380

Figure 32. Observe Mechanism

2381 **11.4.2.1 RETRIEVE request with observe indication**

2382 The Client transmits a RETRIEVE request message to the Server to request updates for the
 2383 Resource on the Server if there is a state change. The RETRIEVE request message carries the
 2384 following parameters:

- 2385 • *fr*: Unique identifier of the Client
- 2386 • *to*: Resource that the Client is requesting to observe
- 2387 • *ri*: Identifier of the RETRIEVE request
- 2388 • *op*: RETRIEVE
- 2389 • *obs*: Indication for observe request

2390 **11.4.2.2 Processing by the Server**

2391 Following the receipt of the RETRIEVE request, the Server may validate if the Client has the
 2392 appropriate rights for the requested operation and the properties are readable and observable. If
 2393 the validation is successful, the Server caches the information related to the observe request. The
 2394 Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial
 2395 response and future responses in case of a change of state.

2396 **11.4.2.3 RETRIEVE response with observe indication**

2397 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request
 2398 message from a Client. The RETRIEVE response message shall include the following parameters.
 2399 If validation succeeded, the response includes an observe indication. If not, the observe indication
 2400 is omitted from the response which signals to the requesting client that registration for notification
 2401 was not allowed.

2402 The RETRIEVE response message shall include the following parameters:

- 2403 • *fr*: Unique identifier of the Server
- 2404 • *to*: Unique identifier of the Client
- 2405 • *ri*: Identifier included in the RETRIEVE request
- 2406 • *cn*: Information resource representation as requested by the Client

- *rs*: The result of the RETRIEVE operation
- *obs*: Indication that the response is made to an observe request

2409 **11.4.2.4 Resource monitoring by the Server**

2410 The Server shall monitor the state the Resource identified in the observe request from the Client.
 2411 Anytime there is a change in the state of the observed resource, the Server sends another
 2412 RETRIEVE response with the observe indication. The mechanism does not allow the client to
 2413 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

2414 **11.4.2.5 Additional RETRIEVE responses with observe indication**

2415 The Server shall transmit updated RETRIEVE response messages following observed changes in
 2416 the state of the Resources indicated by the Client. The RETRIEVE response message shall include
 2417 the parameters listed in section 11.4.2.3.

2418 **11.4.2.6 Cancelling Observe**

2419 The Client can explicitly cancel observe by sending a RETRIEVE request without the observe
 2420 indication field to the same resource on Server which it was observing. For certain protocol
 2421 mappings, the client may also be able to cancel an observe by ceasing to respond to the
 2422 RETRIEVE responses.

2423 **11.5 Device management**

2424 The Device Management includes the following functions:

- Diagnostics and maintenance

2426 The device management functionalities specified in this version of specification are intended to
 2427 address the basic device management features. Addition of new device management features in
 2428 the future versions of the specification is expected.

2429 **11.5.1 Diagnostics and maintenance**

2430 The Diagnostics and Maintenance function in the Framework is intended for use by the
 2431 administrators to resolve issues encountered with the Devices while operating in the field. If
 2432 diagnostics and maintenance is supported by a Device, the Core Resource ‘/oic/mnt’ shall be
 2433 supported as described in Table 20.

2434 **Table 20. Optional diagnostics and maintenance device management Core Resources**

Pre-defined URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
/oic/mnt	Maintenance	oic.wk.mnt	oic.if.rw	The resource through which the device is maintained and can be used for diagnostic purposes. The resource properties exposed by /oic/mnt are listed in Table 21.	Device Management

2435
 2436 Table 21 defines the oic.wk.mnt resource type. At least one of the Factory_Reset, and Reboot
 2437 properties shall be implemented.

2438 **Table 21. 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	no	When writing to this Property: 0 – No action (Default*) 1 – Start Factory Reset After factory reset, this value shall be changed back to the default value (i.e., 0). After factory reset all configuration and state data will be lost. When reading this Property, a value of “1” indicates a pending factory reset, otherwise the value shall be “0” after the factory reset.
Reboot	rb	boolean			R, W	no	When writing to this Property: 0 – No action (Default) 1 – Start Reboot After Reboot, this value shall be changed back to the default value (i.e., 0)

2439

2440 Note: * - Default indicates the value of this property as soon as the device is rebooted or factory reset

2441

2442 The Framework specifies the following commands to be executed on the designated diagnostic
2443 resource of Devices over the network:

- 2444 • Factory_Reset: Updates the device configuration to its original (default) state (factory state
2445 and equivalent to hard reboot)
- 2446 • Reboot: Triggers a soft reboot of a Device maintaining most of the configurations intact

2447 Execution of these commands may result in a change in the configuration state of a Device. The
2448 configuration information in the configuration resource is expected to be updated following
2449 execution of these commands by the Device, if needed. A Client invokes operations on the Server
2450 for executing the Diagnostic functions by sending an UPDATE message to the Server.

2451

2452 11.6 Scenes

2453 11.6.1 Introduction

2454 Scenes are a mechanism for automating certain operations.

2455 A scene is a static entity that stores a set of defined resource property values for a collection of
2456 resources. Scenes provide a mechanism to store a setting over multiple Resources that may be
2457 hosted by multiple separate Servers. Scenes, once set up, can be used by multiple Clients to recall
2458 a setup.

2459 Scenes can be grouped and reused, a group of scenes is also a scene.

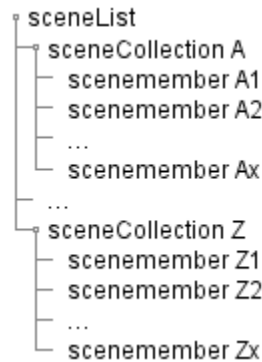
2460 In short, scenes are bundled user settings.

2461 **11.6.2 Scenes**

2462 **11.6.2.1 Introduction**

2463 Scenes are described by means of resources. The scene resources are hosted by a Server and
2464 the top level resource is listed in /oic/res. This means that a Client can determine if the scene
2465 functionality is hosted on a Server via a RETRIEVE on /oic/res or via Resource discovery. The
2466 setup of scenes is driven by Client interactions. This includes creating new scenes, and mappings
2467 of Server resource properties that are part of a scene.

2468 The scene functionality is created by multiple resources and has the structure depicted in Figure
2469 33. The sceneList and sceneCollection resources are overloaded collection resources. The
2470 sceneCollection contains a list of scenes. This list contains zero or more scenes. The
2471 sceneMember resource contains the mapping between a scene and what needs to happen
2472 according to that scene on an indicated resource.

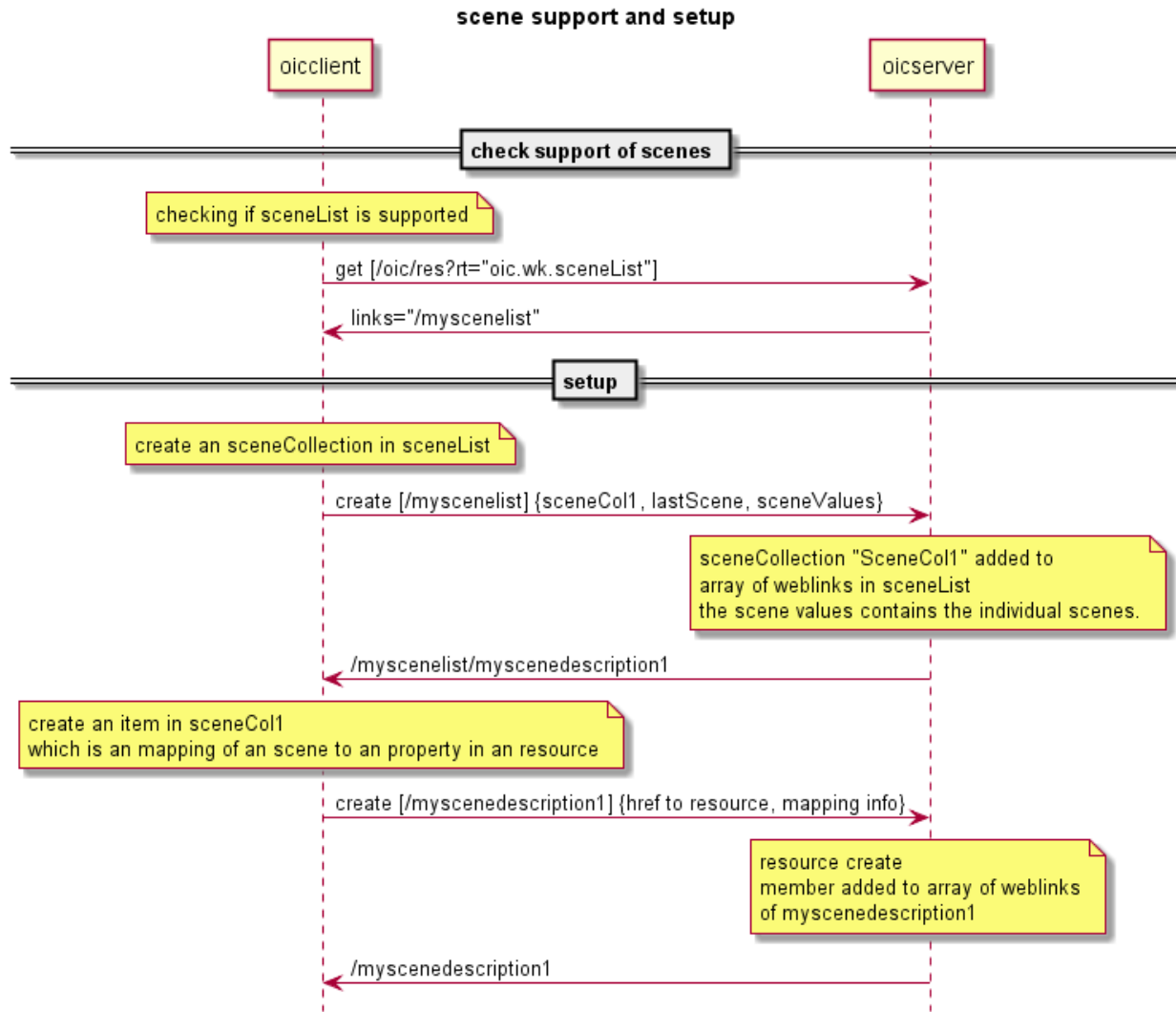


2473

2474 **Figure 33 Generic scene resource structure**

2475 **11.6.2.2 Scene creation**

2476 A Client desiring to interact with scenes needs to first determine if the server supports the scene
2477 feature; the sceneMembers of a scene do not have to be co-located on the server supporting the
2478 scene feature. This can be done by checking if /oic/res contains the rt of the sceneList resource.
2479 This is depicted in first steps of Figure 34. The sceneCollection is created by the Server using
2480 some out of bound mechanism, Client creation of scenes is not supported at this time. This will
2481 entail defining the scene with an applicable list of scene values and the mappings for each
2482 Resource being part of the scene. The mapping for each resource being part of the sceneCollection
2483 is described by a resource called sceneMember. The sceneMember resource contains the link to
2484 a resource and the mapping between the scene listed in the sceneValues property and the actual
2485 resource property value of the Resource indicated by the link.



2486

2487

Figure 34 Interactions to check Scene support and setup of specific scenes

2488

11.6.2.3 Interacting with Scenes

2489

2490

2491

2492

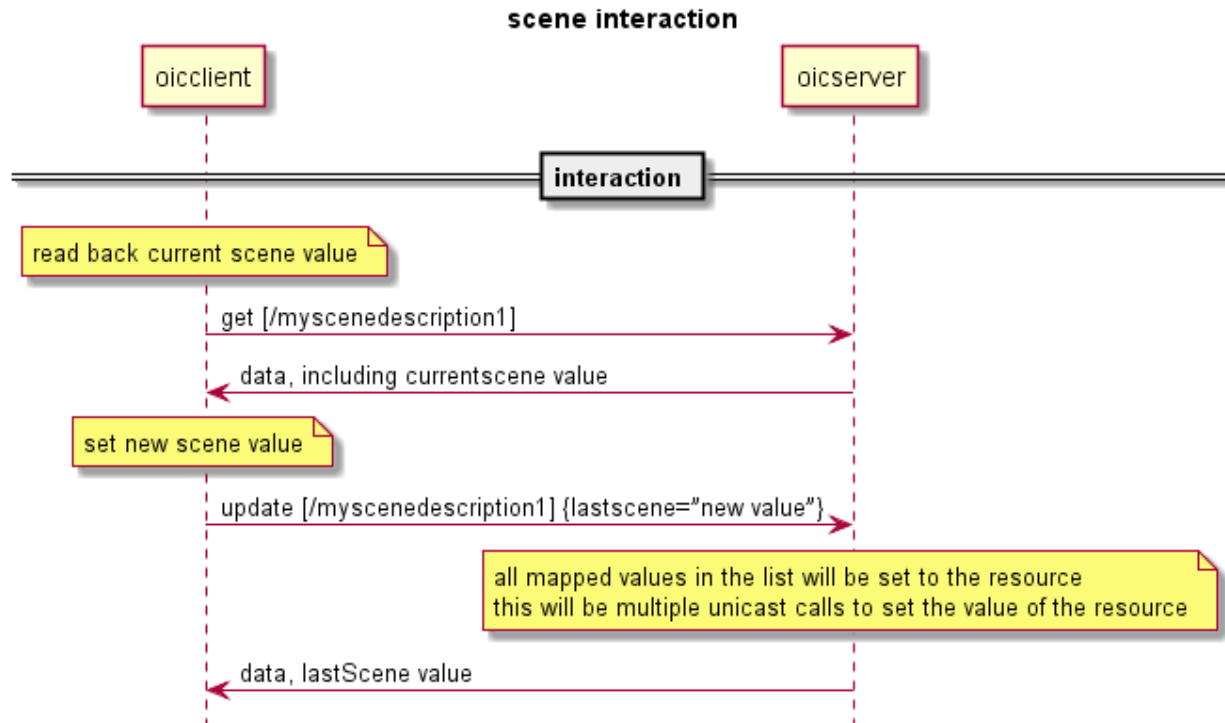
2493

2494

2495

2496

All capable Clients can interact with scenes. The allowed scene values and the last applied scene value can be retrieved from the server hosting the scene. The scene value shall be changed by issuing an UPDATE operation with a payload that sets the lastScene property to one of the listed allowed scene values. These steps are depicted in Figure 35. Note that the lastScene value does not imply that the current state of all resources that are part of the scene will be at the mapped value. This is due to that the setting the scene values are not modelled as actual states of the system. This means that another Client can change just one resource being part of the scene without having feedback that the state of the scene is changed.

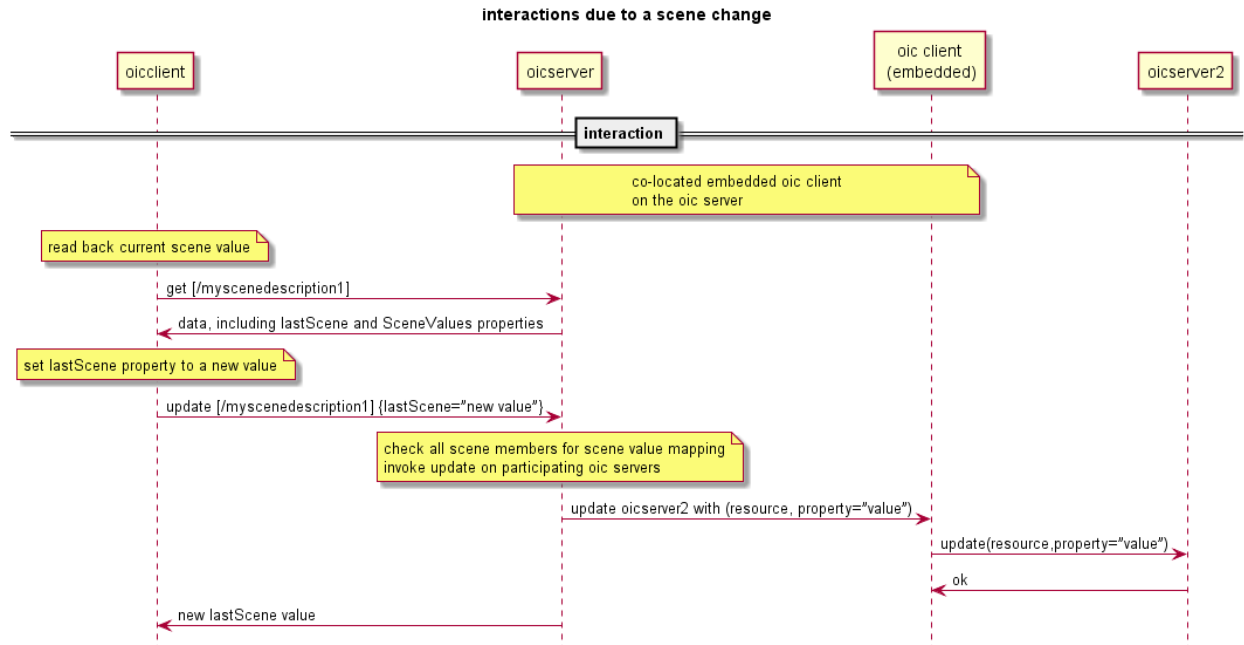


2497

2498

Figure 35 Client interactions on a specific scene

2499 As described previously, a scene can reference one or more resources that are present on one or
 2500 more Servers. The scene members are re-evaluated each time a scene change takes place. This
 2501 evaluation is triggered by a Client that is either embedded as part of the Server hosting the scene,
 2502 or separate to the server having knowledge of the scene via a RETRIEVE operation, observing the
 2503 referenced resources using the mechanism described in section 11.4.2. During the evaluation the
 2504 mappings for the new scene value will be applied to the Server. This behaviour is depicted in
 2505 Figure 36.



2506

2507

Figure 36 Interaction overview due to a Scene change

2508

11.6.2.4 Summary of resource types defined for Scene functionality

2509

Table 22 summarizes the list of resource types that are part of Scenes.

2510

Table 22 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	

2511

11.6.3 Security considerations

2512

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

2513

2514

2515

2516

2517

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

2518

2519

2520

2521 12 Messaging

2522 12.1 Introduction

2523 This section specifies the protocol messaging mapping to the CRUDN messaging operations
2524 (Section 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols
2525 is expected in later version of this specification. All the property information from the resource
2526 model shall be carried within the message payload. This payload shall be generated in the resource
2527 model layer and shall be encapsulated in the data connectivity layer. The message header shall
2528 only be used to describe the message payload (e.g., verb, mime-type, message payload format),
2529 in addition to the mandatory header fields defined in messaging protocol (e.g., CoAP) specification.
2530 If the message header does not support this, then this information shall also be carried in the
2531 message payload. Resource model information shall not be included in the message header
2532 structure unless the message header field is mandatory in the messaging protocol specification.

2533 12.2 Mapping of CRUDN to CoAP

2534 12.2.1 Overview

2535 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in section
2536 12.2.3. A Device implementing CoAP shall conform to IETF draft-ietf-core-observe-16 to
2537 implement the CoAP Observe option. Support for CoAP block transfer when the payload is larger
2538 than the MTU is defined in section 12.2.6.

2539 12.2.2 URIs

2540 An OCF: URI is mapped to a coap: URI by replacing the scheme name 'oic' with 'coap' if unsecure
2541 or 'coaps' if secure before sending over the network by the requestor. Similarly on the receiver
2542 side, the scheme name is replaced with 'oic'.

2543 12.2.3 CoAP method with request and response

2544 12.2.3.1 Overview

2545 Every request has a CoAP method that realizes the request. The primary methods and their
2546 meanings are shown in Table 23, which provides the mapping of GET/PUT/POST/DELETE
2547 methods to CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides
2548 the generic behaviours when using these methods, however resource interfaces may modify these
2549 generic semantics.
2550

2551 **Table 23. CoAP request and response**

Method for CRUDN	(mandatory) Request data	(mandatory) Response data
GET for RETRIEVE	- Method code: GET (0.01) - Request URI: an existing URI for the Resource to be retrieved	- Response code: success (2.xx) or error (4.xx) - Payload: Resource representation of the target Resource (when successful)
POST for CREATE	- Method code: POST (0.02) - Request URI: an existing URI for the Resource responsible for the creation - Payload: Resource presentation of the Resource to be created	- Response code: success (2.xx) or error (4.xx) - Payload: the URI of the newly created Resource (when successful).
PUT for CREATE	- Method code: PUT (0.03) - Request URI: a new URI for the Resource to be created. - Payload: Resource presentation of the Resource to be created.	- Response code: success (2.xx) or error (4.xx)
POST for UPDATE	- Method code: POST (0.02)	- Response Code: success (2.xx) or error (4.xx)

	- Request URI: an existing URI for the Resource to be updated. - Payload: representation of the Resource to be updated.	
DELETE for DELETE	- Method code: DELETE (0.04) - Request URI: an existing URI for the Resource to be deleted.	- Response code: success (2.xx) or error (4.x)

2552

2553 **12.2.3.2 CREATE with POST or PUT**

2554 **12.2.3.2.1 With POST**

2555 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
2556 existing Resource on the Server that is processing the request. If no such Resource is present,
2557 the Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall
2558 use an existing request URI which identifies the Resource on the Server responsible for creation.
2559 The URI of the created Resource is determined by the Server and provided to the Client in the
2560 response.

2561 A Client shall include the representation of the new Resource in the request payload. The new
2562 resource representation in the payload shall have all the necessary properties to create a valid
2563 Resource instance, i.e. the created Resource should be able to properly respond to the valid
2564 Request with mandatory Interface (e.g., GET with ?if=oic.if.baseline).

2565 Upon receiving the POST request, the Server shall either

- 2566 • create the new Resource with a new URI, respond with the new URI for the newly created
2567 Resource and a success response code (2.xx); or
- 2568 • respond with an error response code (4.xx).

2569 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or
2570 guaranteed.

2571 **12.2.3.2.2 With PUT**

2572 PUT shall be used to create a new Resource or completely replace the entire representation of an
2573 existing Resource. The resource representation in the payload of the PUT request shall be the
2574 complete representation. PUT for CREATE shall use a new request URI identifying the new
2575 Resource to be created.

2576 The new resource representation in the payload shall have all the necessary properties to create
2577 a valid Resource instance, i.e. the created Resource should be able to properly respond to the
2578 valid Request with mandatory Interface (e.g. GET with ?if=oic.if.baseline).

2579 Upon receiving the PUT request, the Server shall either

- 2580 • create the new Resource with the request URI provided in the PUT request and send back a
2581 response with a success response code (2.xx); or
- 2582 • respond with an error response code (4.xx).

2583 PUT is an unsafe method but it is idempotent, thus when a PUT request is repeated the outcome
2584 is the same each time.

2585 **12.2.3.3 RETRIEVE with GET**

2586 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of
2587 the target Resource identified by the request URI.

2588 Upon receiving the GET request, the Server shall either

2589 • send back the response with the representation of the target Resource with a success response
2590 code (2.xx); or

2591 • respond with an error response code (4.xx) or ignore it (e.g. non-applicable multicast GET).

2592 GET is a safe method and is idempotent.

2593 **12.2.3.4 UPDATE with POST**

2594 POST shall be used only in situations where the request URI is valid, that is it is the URI of an
2595 existing Resource on the Server that is processing the request. If no such Resource is present,
2596 the Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE
2597 Property values of an existing Resource (see Sections 3.1.32 and 8.4.2).

2598 Upon receiving the request, the Server shall either

- 2599 • apply the request to the Resource identified by the request URI in accordance with the applied
2600 interface (i.e. POST for non-existent Properties is ignored) and send back a response with a
2601 success response code (2.xx); or
- 2602 • respond with an error response code (4.xx). Note that If the representation in the payload is
2603 incompatible with the target Resource for POST using the applied interface (i.e. the "overwrite"
2604 semantic cannot be honored because of read-only property in the payload), then the error
2605 response code 4.xx shall be returned.

2606 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or
2607 guaranteed.

2608 **12.2.3.5 DELETE with DELETE**

2609 DELETE shall be used for DELETE operation. The DELETE method requests that the resource
2610 identified by the request URI be deleted.

2611 Upon receiving the DELETE request, the Server shall either

- 2612 • delete the target Resource and send back a response with a success response code (2.xx); or
- 2613 • respond with an error response code (4.xx).

2614 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

2615
2616

2617 **12.2.4 Content Type negotiation**

2618 The Device framework mandates support of CBOR, however it allows for negotiation of the payload
2619 body if more than one encoding type is supported by an implementation. In this case the accept
2620 option defined in section 5.10.4 of IETF RFC 7252 shall be used to indicate which content
2621 encodings are requested by the Client.

2622 Content types supported are as shown in Table 24.

2623

Table 24. 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

2624 Note: An OCF vertical can mandate a specific content type.

2625 Server and Client shall send a Content-Format option every time in a message with a payload
2626 body. The Content Format option shall use the Content Format numeric value from Table 24.

2627 **12.2.5 CRUDN to CoAP response codes**

2628 The mapping of CRUDN operations response codes to CoAP response codes are identical to the
2629 response codes defined in IETF RFC 7252.

2630 **12.2.6 CoAP block transfer**

2631 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT
2632 devices. However scenarios can be envisioned in which an application needs to transfer larger
2633 payloads.

2634 CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be used by all Servers
2635 which generate a content payload that would exceed the size of a CoAP datagram as the result of
2636 handling any defined CRUDN operation.

2637 Similarly, CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be supported
2638 by all Clients. The use of block-wise transfer is applied to both the reception of payloads as well
2639 as transmission of payloads that would exceed the size of a CoAP datagram.

2640 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the
2641 same reliability setting (i.e. all confirmable or all non-confirmable).

2642 A Client may support both the block1 (as descriptive) and block2 (as control) options as described
2643 by IETF draft-ietf-core-block-18. A Server may support both the block1 (as control) and block2 (as
2644 descriptive) options as described by IETF draft-ietf-core-block-18.

2645 **12.2.7 CoAP serialization over TCP**

2646 **12.2.7.1 Introduction**

2647 In environments where TCP is already available, CoAP can take advantage of it to provide
2648 reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For
2649 example, consider a cloud application acting as a Client and the Server is located at the user's
2650 home. The Server which already support CoAP as a messaging protocol (e.g., Smart Home vertical
2651 profile) could easily support CoAP serialization over TCP rather than adding another messaging
2652 protocol. A Device implementing CoAP Serialization over TCP shall conform to IETF draft-
2653 tschofenig-core-coap-tcp-tls-04.

2654 **12.2.7.2 Indication of support**

2655 If UDP is blocked, clients depend on the pre-configured details on the device to find support for
2656 CoAP over TCP. If UDP is not-blocked, a Device which supports CoAP serialization over TCP shall
2657 populate the Messaging Protocol (mpro) property in oic/res with the value "coap+tcp" or "coaps+tcp"
2658 to indicate that the device supports messaging protocol as specified by section 11.3.4.

2659 **12.2.7.3 Message type and header**

2660 The message type transported between Client and Server shall be a non-confirmable message
 2661 (NON). The protocol stack used in this scenario shall be as described in section 3 in IETF draft-
 2662 tschofenig-core-coap-tls-04.

2663 The CoAP header as described in figure 6 in IETF draft-tschofenig-core-coap-tls-04 shall be
 2664 used for messages transmitted between a Client and a Server. A Device shall use “Alternative L3”
 2665 as defined in IETF draft-tschofenig-core-coap-tls-04.

2666 **12.2.7.4 URI scheme**

2667 The URI scheme used shall be as defined in section 6 in IETF draft-tschofenig-core-coap-tls-
 2668 04].

2669 For the “coaps+tcp” URI scheme the “TLS Application Layer Protocol Negotiation Extension”
 2670 IETF RFC 7301 shall be used.

2671 **12.2.7.5 KeepAlive**

2672 **12.2.7.5.1 Overview**

2673 In order to ensure that the connection between a Device is maintained, when using CoAP
 2674 serialization over TCP, a Device that initiated the connection should send application layer
 2675 KeepAlive messages. The reasons to support application layer KeepAlive are as follows:

- 2676 • TCP KeepAlive only guarantees that a connection is alive at the network layer, but not at the
 2677 application layer
- 2678 • Interval of TCP KeepAlive is configurable only using kernel parameters, and is OS dependent
 2679 (e.g., 2 hours by default in Linux)

2680 **12.2.7.5.2 KeepAlive Mechanism**

2681 Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. A Server shall
 2682 support a resource of type oic.wk.ping as defined in Table 25.

2683 **Table 25. Ping resource**

Pre-defined URI	Resource Type Title	Resource Type ID (“rt” value)	Interfaces	Description	Related Functional Interaction
/oic/ping	Ping	oic.wk.ping	oic.if.rw	The resource using which a Client keeps its Connection with a Server active. The resource properties exposed by /oic/ping are listed in Table 26.	KeepAlive

2684
 2685 Table 26 defines oic.wk.ping resource type.

2686 **Table 26. 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.

2687 The following steps detail the KeepAlive mechanisms for a Client and Server:

- 2688 1) A Client which wants to keep the connection with a Server alive shall send a PUT request to
2689 /oic/ping resource on the Server updating its connection Interval.
- 2690 a) This time interval shall start from 2 minutes and increases in multiples of 2 up to a maximum
2691 of 64 minutes. It stays at 64 minutes from that point.
- 2692 2) An Server receiving this ping request shall respond within 1 minute.
- 2693 3) If a Client does not receive the response within 1 minute, it shall terminate the connection.
- 2694 4) If an Server does not receive a PUT request to ping resource within the specified "interval"
2695 time, the Server shall terminate the connection.

2696 An example of the KeepAlive mechanism is as follows:

- 2697 • Client → Server: PUT /oic/ping {interval: 2}
- 2698 • Server → Client: 2.03 valid

2699

2700 **12.3 Payload Encoding in CBOR**

2701 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to
2702 JSON from CBOR in accordance with IETF RFC 7049 section 4 unless otherwise specified in this
2703 section.

2704 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types
2705 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-
2706 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation
2707 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer
2708 numbers shall be within the open range $(-2^{53}, 2^{53})$. Properties defined as a JSON number
2709 should be encoded as integers whenever possible; if this is not possible Properties defined as a
2710 JSON number should use single-precision if the loss of precision does not affect the quality of
2711 service, otherwise the Property shall use double-precision.

2712

2713 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values
2714 in any position. If a property defined as a JSON integer is received encoded other than as an
2715 integer, the implementation may reject this encoding using a final response as appropriate for the
2716 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a property is
2717 defined as a JSON number an implementation shall accept integers, single- and double-precision
2718 floating point.

2719 **13 Security**

2720 The details for handling security and privacy are specified in [OCF Security].

2721

2722 **14 Multi resource model support**

2723 **14.1 Interoperability issue**

2724 **14.1.1 Multiple IoT Standards**

2725 Note: Alignment and interoperability between models will be added in a later version of the
2726 specification.

2727 IoT requires standardization for interoperability among diverse devices and multiple standards are
2728 under development currently. IETF defines network and web transfer protocol (e.g. 6lowpan
2729 [RFC6775] and CoAP [RFC6690], [RFC7252]), oneM2M [oneM2M] produces technical

2730 specifications for a common M2M Service Layer [oneM2M-TS0001], [oneM2M-TS0004] and IPSO
2731 Alliance [IPSO] publishes Smart Object Guideline [IPSOSmartObjects].

2732 Multitude of IoT standards are based on "Representational State Transfer (REST)", which is a
2733 software architecture style with a coordinated set of constraints for the design of components in a
2734 distributed hypermedia system [REST]. In REST based IoT, a real world entity is represented as
2735 resource in a server, which a client accesses and manipulates the resource through
2736 representations to interact with the entity, i.e. sensing and controlling the physical environments.
2737 Moreover several IoT standards adopt the common network and web transfer protocols. oneM2M,
2738 IPSO and OCF all use CoAP and IP/ UDP, [oneM2M-TS0008], [IPSO], [OCF] so any client and
2739 server supporting those standards can exchange request and response messages.

2740 However in order to interact properly, it's not sufficient for IoT devices to be able to transfer CoAP
2741 messages. IoT devices should understand each other's resources and be aware of their semantic
2742 meaning and syntactic form. Currently each standard defines its own "resource model" and
2743 specifies a different scheme to construct resources from physical entities such as light [OCF],
2744 [IPSOFramework], [IPSOSmartObjects], [oneM2M-TS0001]. Hence client and server adopting
2745 different standards can't perform meaningful interaction, i.e. the client can't manipulate the
2746 resource representation in the server.

2747 For wider interoperability among multiple standards, IoT devices need to understand each other's
2748 resource model to process CoAP request and response message properly. To interpret resources
2749 correctly, client and server need to determine which resource model each other follows in the first
2750 place. The client should be aware of whether its corresponding server adopts oneM2M or OCF
2751 model and vice versa.

2752 **14.1.2 Different resource models**

2753 OCF specification follows a resource oriented architecture with RESTful architectural style.
2754 Without common understanding on resource model, two IoT devices can't interact with each other.

2755 Currently multiple organizations such as OCF, IPSO Alliance or oneM2M, define their own resource
2756 model in difference ways, which may restrict interoperability to the respective ecosystems. The
2757 main discrepancies are as follows

- 2758 • **Resource structure:** Some define resource to have attributes (e.g. oneM2M), whereas
2759 others define it atomic and not decomposed into attributes (e.g. IPSO alliance). For
2760 example, a smart light may be represented as a resource with on-off attribute or a
2761 resourcecollection with on-off resource. In the former, on-off attribute doesn't have URI
2762 and should be accessed indirectly via the resource. In the latter, being a resource itself,
2763 on-off resource is assigned its own URI and can be directly manipulated.
- 2764 • **Resource name & type:** Some allow resource to be named freely and indicate its
2765 characteristic with separate resource type attribute (e.g. oneM2M). Whereas others fix the
2766 name of resource a priori and indicate its characteristic with the name itself (e.g.
2767 IPSOalliance). For example, smart light can be named anyway such as 'LivingRoomLight_1'
2768 in oneM2M but should have the fixed Object name with numerical Object ID of "IPSO Light
2769 Control (3311)" in IPSO alliance. Furthermore, in consequence, it's likely that data path in
2770 URI is freely defined in the former and predetermined for the latter.
- 2771 • **Resource hierarchy:** Some allow resource to be organized in hierarchy so that resource
2772 includes another resource in itself with parent-child relationship (e.g. oneM2M). Whereas
2773 others mandate resource to be of flat structure and associate with other resources only by
2774 referencing their links.

2775 In addition to the above, different organizations use different syntax and have different features
2776 (e.g. resource interface), which will inhibit IoT interoperability. When IoT client and server don't
2777 understand the resource model each supports, they can't perform RESTful transaction.

2778 For example, a smart light can be represented as an IPSO Smart Object in JSON as below:

2779

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

2780

2781

2782 In the above, "3311" is an "Object ID" defining object type, 0" an "Object Instance", designating
2783 one or more instances, "5850", "5851", "Resource ID", defining resource type. Also IPSO embeds
2784 resource information in data path, so "On/Off" resource has predetermined data path of
2785 "3311/0/5850" and "Dimmer" resource datapath of "3311/0/5851"

2786

2787 Whereas the same smart light may be represented in OCF as two Resources.

2788

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

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

2789

2790

2791 **14.2 A scheme to exchange resource model information**

2792 **14.2.1 A scheme to exchange resource model information**

2793 IoT devices, i.e. client and server, need to understand the resource model which their
2794 corresponding device supports to be able to interoperate each other.

2795 For the initial step, it would help for IoT devices to indicate resource model each device supports.
2796 Then client and server may choose a common resource model for interaction, or in the absence of
2797 such a common model, rely on translation between the models, possibly with the assistance of 3rd
2798 party such as intermediary. Alignment and interoperability between models will be added in a later
2799 version of the specification.

2800 This document presents a scheme for CoAP endpoints, client and server, to exchange resource
2801 model they support.

2802 First, the Internet media type and Content-Format identifier are used to indicate a specific resource
2803 model. The Internet media types can be defined to indicate the resource models, potentially with
2804 content-coding, such as "application/ips+json", then assigned numeric Content-Format identifiers
2805 such as "123123" to minimize payload overhead for CoAP usage.

2806 Second, CoAP Accept and Content-Format Option are used to exchange the Content-Format
2807 identifiers indicating the resource models which CoAP endpoints prefer or support. A client
2808 includes the CoAP Accept option to inform a server which resource model, potentially with content-
2809 encoding, is acceptable and the server returns the payload in the preferred resource model if
2810 available. The Content-Format Option indicates the resource model which the payload follows.

2811

2812
2813
2814
2815

Annex A (informative)

Operation Examples

2816 A.1 Introduction

2817 This section describes some example scenarios using sequence of operations between the entities
2818 involved. In all the examples below “Light” is a Server and “Smartphone” is a Client. In one of the
2819 scenario “Garage” additionally acts as a Server. All the examples are based on the following
2820 example resource definitions:

2821 `rt=oc.example.light` with resource type definition as illustration in Table 27.

2822 **Table 27. `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

2823

2824 `rt=oc.example.garagedoor` with resource type definition as illustration in Table 28.

2825 **Table 28. `oc.example.garagedoor` resource type definition**

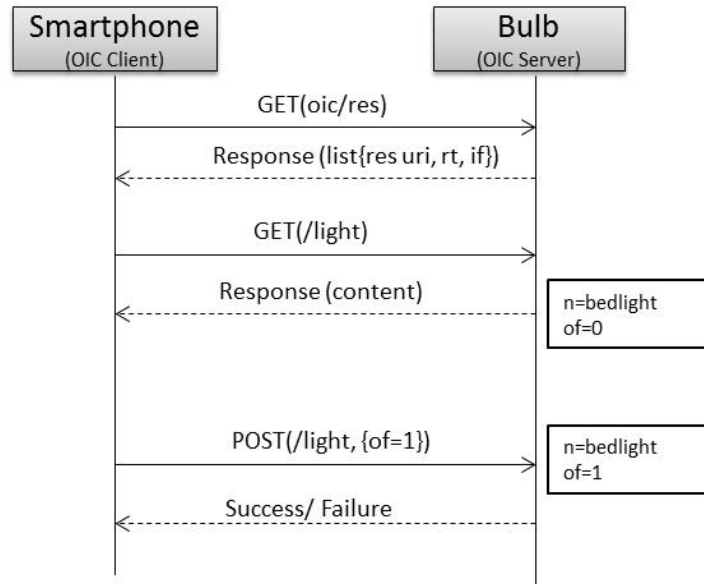
Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	n	string			R, W	no	
open-close	oc	boolean			R, W	yes	Open/Close Control: 0 = Open 1 = Close

2826

2827 `/oic/mnt (rt=oc.wk.mnt)` used in below examples is defined in section 11.5.1.

2828 A.2 When at home: From smartphone turn on a single light

2829 This sequence highlights (Figure 37) the discovery and control of an OCF light resource from an
2830 OCF smartphone.



2831

2832

Figure 37. When at home: from smartphone turn on a single light

2833 Discovery request can be sent to “All OCF Nodes” Multicast address FF0X::158 or can be sent
2834 directly to the IP address of device hosting the light resource.

- 2835 1) Smartphone sends a GET request to /oic/res resource to discover all resources hosted on
2836 targeted end point
- 2837 2) The end point (bulb) responds with the list of resource URI, resource type and interfaces
2838 supported on the end point (one of the resource is '/light' whose rt=oc.example.light)
- 2839 3) Smartphone sends a GET request to '/light' resource to know its current state
- 2840 4) The end point responds with representation of light resource ({n=bedlight;of=0})
- 2841 5) Smartphone changes the 'of' property of the light resource by sending a POST request to '/light'
2842 resource ({of=1})
- 2843 6) On Successful execution of the request, the end point responds with the changed resource
2844 representation. Else, error code is returned. Details of the error codes are defined in section
2845 12.2.5.

2846 **A.3 GroupAction execution**

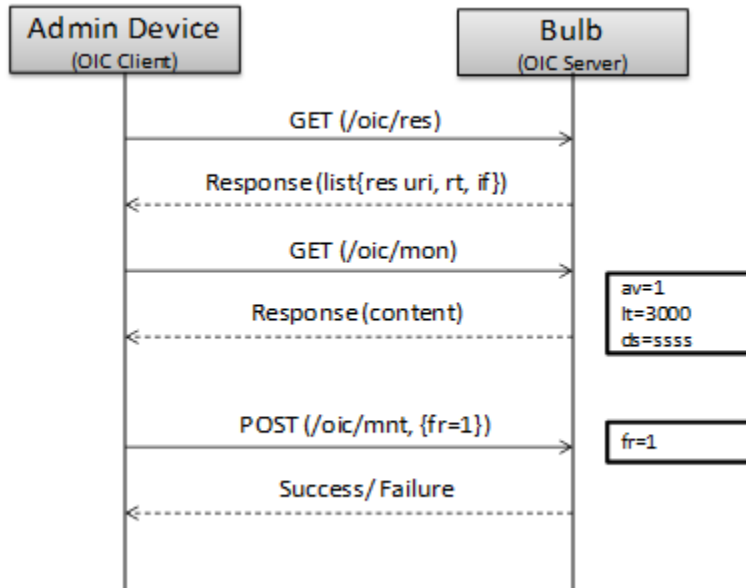
2847 This example will be added when groups feature is added in later version of specification

2848 **A.4 When garage door opens, turn on lights in hall; also notify smartphone**

2849 This example will be added when scripts feature is added in later version of specification

2850 **A.5 Device management**

2851 This sequence highlights (Figure 38) the device management function of maintenance.



2853

2854

Figure 38. Device management (maintenance)

2855 **Pre-Condition:** Admin device has different security permissions and hence can perform device
2856 management operations on the Device

- 2857 1) Admin device sends a GET request to `/oic/res` resource to discover all resources hosted on a
2858 targeted end point (in this case Bulb)
- 2859 2) The end point (bulb) responds with the list of resource URI, resource type and interfaces
2860 supported on the end point (one of the resources is `/oic/mnt` whose `rt=oc.wk.mnt`)
- 2861 3) Admin Device changes the 'fr' property of the maintenance resource by sending a POST
2862 request to `/oic/mnt` resource (`{fr=1}`). This triggers a factory reset of the end point (bulb)
- 2863 4) On successful execution of the request, the end point responds with the changed resource
2864 representation. Else, error code is returned. Details of the error codes are defined in section
2865 12.2.5.

2866
2867
2868
2869

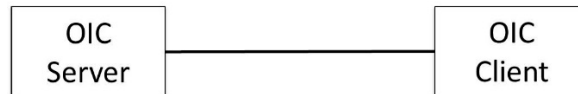
Annex B (informative)

OCF interaction scenarios and deployment models

2870 B.1 OCF interaction scenarios

2871 A Client connects to one or multiple Servers in order to access the resources provided by those
2872 Servers. The following are scenarios representing possible interactions among Roles:

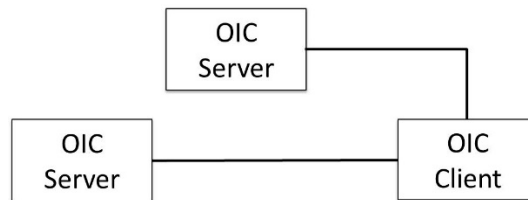
- 2873 • Direct interaction between Client and Server (Figure 39). In this scenario the Client and the
2874 Server directly communicate without involvement of any other Device. A smartphone which
2875 controls an actuator directly uses this scenario.



2876

2877 **Figure 39. Direct interaction between Server and Client**

- 2878 • Interaction between Client and Server using another server (Figure 40). In this scenario,
2879 another Server provides the support needed for the Client to directly access the desired
2880 resource on a specific Server. This scenario is used for example, when a smartphone first
2881 accesses a discovery server to find the addressing information of a specific appliance, and
2882 then directly accesses the appliance to control it.



2883

2884 **Figure 40. Interaction between Client and Server using another Server**

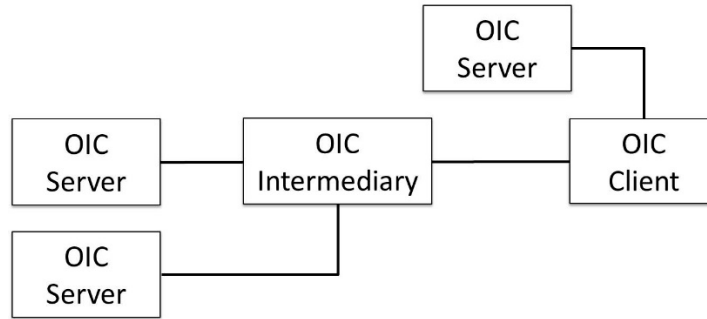
- 2885 • Interaction between Client and Server using Intermediary (Figure 41). In this scenario an
2886 Intermediary facilitates the interaction between the Client and the Server. A smartphone which
2887 controls appliances in a smart home via MQTT broker uses this scenario.



2888

2889 **Figure 41. Interaction between Client and Server using Intermediary**

- 2890 • Interaction between Client and Server using support from multiple Servers and intermediary
2891 (Figure 42). In this scenario, both Server and Intermediary roles are present to facilitate the
2892 transaction between the Client and a specific Server. An example scenario is when a
2893 smartphone first accesses a Resource Directory (RD) server to find the address to a specific
2894 appliance, then utilizes MQTT broker to deliver a command message to the appliance. The
2895 smartphone can utilize the mechanisms defined in CoRE Resource Directory such as default
2896 location, anycast address or DHCP (IETF draft-ietf-core-resource-directory-02) to discover the
2897 Resource Directory information.

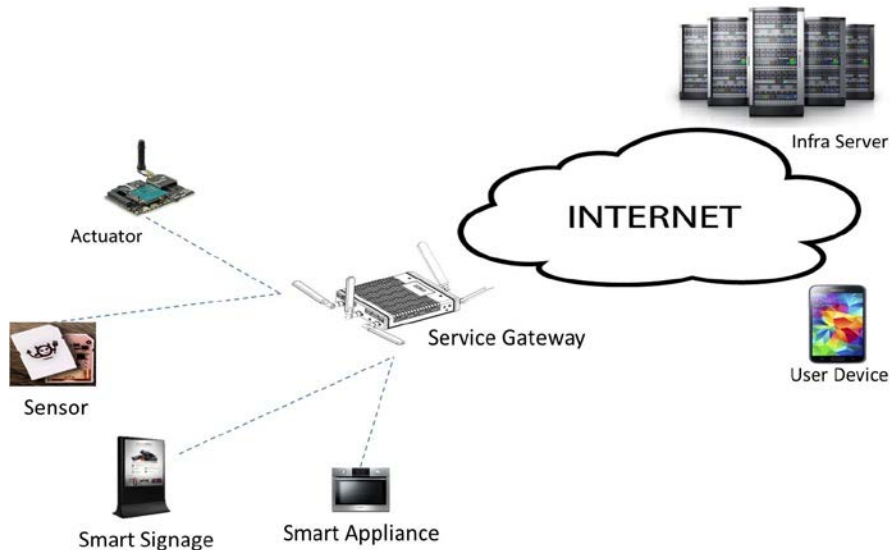


2898

2899 **Figure 42. Interaction between Client and Server using support from multiple Servers and**
 2900 **Intermediary**

2901 **B.2 Deployment model**

2902 In deployment, Devices are deployed and interact via either wired or wireless connections. Devices
 2903 are the physical entities that may host resources and play one or more Roles. There is no constraint
 2904 on the structure of a deployment or number of Devices in it. Architecture is flexible and scalable
 2905 and capable of addressing large number of devices with different device capabilities, including
 2906 constrained devices which have limited memory and capabilities. Constrained devices are defined
 2907 and categorized in [TCNN].



2908

2909 **Figure 43. Example of Devices**

2910 Figure 43 depicts a typical deployment and set of Devices, which may be divided in the following
 2911 categories:

- 2912 • **Things:** Networked devices which are able to interface with physical environments. Things are
 2913 the devices which are primarily controlled and monitored. Examples include smart appliances,
 2914 sensors, and actuators. Things mostly take the role of Server but they may also take the role of
 2915 Client, for example in machine-to-machine communications.
- 2916 • **User Devices:** Devices employed by the users enabling the users to access resources and
 2917 services. Examples include smart phones, tablets, and wearable devices. User Devices mainly
 2918 take the role of Client, but may also take the role of Server or Intermediary.

- 2919 • **Service Gateways:** Network equipment which take the role of Intermediary. Examples are
2920 home gateways.
- 2921 • **Infra Servers:** Data centers residing in cloud infrastructure, which facilitate the interaction
2922 among Devices by providing network services such as AAA, NAT traversal or discovery. It can
2923 also play the role of Client or Intermediary

2924 **Annex C**
2925 (informative)

2926 **Other Resource Models and OCF Mapping**
2927

2928 **C.1 Multiple resource models**

2929 RESTful interactions are defined dependent on the resource model; hence, Devices require a
2930 common understanding of the resource model for interoperability.

2931 There are multiple resource models defined by different organizations including OCF, IPSO
2932 Alliance and oneM2M, and used in the industry, which may restrict interoperability among
2933 respective ecosystems. The main differences from Resource model are as follows:

- 2934 • **Resource structure:** Resources may be defined to have properties (e.g., oneM2M defined
2935 resources), or may be defined as an atomic entity and not be decomposable into properties
2936 (e.g., IPSO alliance defined resources). For example, a smart light may be represented as a
2937 resource with an on-off property or a resource collection containing an on-off resource. In the
2938 former, on-off property doesn't have a URI of its own and can only be accessed indirectly via
2939 the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and
2940 can be directly manipulated.
- 2941 • **Resource name & type:** Resources may be allowed to be named freely and have their
2942 characteristics indicated using a resource type property (e.g., as defined in oneM2M).
2943 Alternatively, the name of resources may be defined a priori in a way that the name by itself is
2944 indicative of its characteristic (e.g., as defined by IPSO alliance). For example, in oneM2M
2945 resource model, a smart light can be named with no restrictions, such as 'LivingRoomLight_1'
2946 but in IPSO alliance resource model it is required to have the fixed Object name with numerical
2947 Object ID of "IPSO Light Control (3311)". Consequently, it's likely that in the former case the
2948 data path in URI is freely defined and in the latter case it is predetermined.
- 2949 • **Resource hierarchy:** Resources may be allowed to be organized in hierarchy where a resource
2950 contains another resource with a parent-child relationship (e.g., in oneM2M definition of
2951 resource model). Resources may also be required to have a flat structure and associate with
2952 other resources only by referencing their links.

2953 In addition to the above, different organizations use different syntax and define different features
2954 (e.g., resource interface), which preclude interoperability.

2955 **C.2 OCF approach for support of multiple resource models**

2956 In order to expand the IoT ecosystem the Framework takes an inclusive approach for interworking
2957 with existing resource models. Specifically, the Framework defines a resource model while
2958 providing a mechanism to easily map to other models. By embracing existing resource models
2959 OCF is inclusive of existing ecosystems while allowing for the transition toward definition of a
2960 comprehensive resource model integrating all ecosystems.

2961 The following OCF characteristics enable support of other resource models:

- 2962 • **resource model is the superset of multiple models:** the resource model is defined as the
2963 superset of existing resource models. In other words, any existing resource model can be
2964 mapped to a subset of resource model concepts.
- 2965 • **Framework may allow for resource model negotiation:** the Client and Server exchange the
2966 information about what resource model(s) each supports. Based on the exchanged information,
2967 the Client and Server choose a resource model to perform RESTful interactions or to perform
2968 translation. This feature is out of scope of the current version of this specification, however,
2969 the following is a high level description for resource model negotiation.

2970 **C.3 Resource model indication**

2971 The Client and server exchange the information about what resource model(s) each supports.
2972 Based on the exchanged information, the Client and Server choose a resource model to perform
2973 RESTful interactions or to perform translation. The exchange could be part of discovery and
2974 negotiation. Based on the exchange, the Client and Server follow a procedure to ensure
2975 interoperability among them. They may choose a common resource model or execute translation
2976 between resource models.

- 2977 • **Resource model schema exchange:** The Client and Server may share the resource model
2978 information when they initiate a RESTful interaction. They may exchange the information about
2979 which resource model they support as part of session establishment procedures. Alternatively,
2980 each request or response message may carry the indication of which resource model it is using.
2981 For example, [COAP] defines “Content-Format option” to indicate the “representation format”
2982 such as “application/json”. It’s possible to extend the Content-Format Option to indicate the
2983 resource model used with the representation format such as “application/ipsso-json”.
- 2984 • **Ensuing procedures:** After the Client and Server exchange the resource model information,
2985 they perform a suitable procedure to ensure interoperability among them. The simplest way is
2986 to choose a resource model supported by both the Client and Server. In case there is no
2987 common resource model, the Client and Server may interact through a 3rd party.

2988 In addition to translation which can be resource intensive, a method based on profiles can be used
2989 in which an OCF implementation can accommodate multiple profiles and hence multiple
2990 ecosystems.

- 2991 • **Resource Model Profile:** the Framework defines resource model profiles and implementers or
2992 users choose the active profile. The chosen profile constraints the Device to strict rules in how
2993 resources are defined, instantiated and interacted with. This would allow for interoperation with
2994 devices from the ecosystem identified by the profile (e.g., IPSO, OneM2M etc.). Although this
2995 enables a Device to participate in and be part of any given ecosystem, this scheme does not
2996 allow for generic interoperability at runtime. While this approach may be suitable for resource
2997 constrained devices, more resource capable devices are expected to support more than one
2998 profile.

2999 **C.4 An Example Profile (IPSO profile)**

3000 IPSO defines smart objects that have specific resources and they take values determined by the
3001 data type of that resource. The smart object specification defines a category of such objects. Each
3002 resource represents a characteristic of the smart object being modelled.

3003 While the terms may be different, there are equivalent concepts in OCF to represent these terms.
3004 This section provides the equivalent OCF terms and then frames the IPSO smart object in OCF
3005 terms.

3006 The IPSO object Light Control defined in Section 16 of the IPSO Smart Objects 1.0 is used as the
3007 reference example.

3008 **C.4.1 Conceptual equivalence**

3009 The IPSO smart object definition is equivalent to an Resource Type definition which defines the
3010 relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent to
3011 a Property that like an IPSO Resource has a defined data type, enumeration of acceptable values,
3012 units, a general description and access modes (based on the Interface).

3013 The general method for developing the equivalent Resource Type from an IPSO Smart Object
3014 definition is to ignore the Object ID and replace the Object URN with and OCF ‘.’ (dot) separated
3015 name that incorporates the IPSO object. Alternatively the Object URN can be used as the Resource

3016 Type ID as is (as long as the URN does not contain any '.' (dots)) – using the same Object URN
 3017 as the Resource Type ID allows for compatibility when interacting with an IPSO compliant device.
 3018 The object URN based naming does not have any bearing for OCF to OCF interoperability and so
 3019 the OCF format is preferred – for OCF to OCF interoperability only the data model consistency is
 3020 required.

3021 Two models are available to render IPSO objects into OCF.

- 3022 1) One is where the IPSO Smart Object represents a Resource. In this case, the IP Smart Object
 3023 is regarded as a resource with the Resource Type matching the description of the Smart Object.
 3024 Furthermore, each resource in the IPSO definition is represented as an Property in the
 3025 Resource Type (the IPSO Resource ID is replaced with a string representing the Property).
 3026 This is the preferred approach when the IPSO Data Model is expressed in the Resource Model.
- 3027 2) The other approach is to model an IPSO Smart Object as an Collection. Each IPSO Resource
 3028 is then modelled as an Resource with an Resource Type that matches the definition of the
 3029 IPSO Resource. Each of these resource instances are then bound to the Collection that
 3030 represents this IPSO Smart Object.

3031

3032 Below is an example showing how an IPSO LightControl Object is modelled as a Resource.

3033 **Resource Type: Light Control**

3034 Description: This Object is used to control a light source, such as a LED or other light. It allows a
 3035 light to be turned on or off and its dimmer setting to be controlled as a percentage value between
 3036 0 and 100. An optional colour setting enables a string to be used to indicate the desired colour.
 3037 Table 29 and Table 30 define the resource type and its properties, respectively.

3038

Table 29. Light control resource type definition

Resource Type	Resource Type ID	Multiple Instances	Description
Light Control	"oic.light.control" or "urn:oma:lwm2m:ext:3311"	Yes	Light control object with on/off and optional dimming and energy monitor

3039

3040

Table 30. 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

3041
3042

3043
3044
3045
3046

Annex D (normative)

Resource Type definitions

3047 D.1 List of resource type definitions

3048 Table 31 contains the list of defined core resources in this specification.

3049 **Table 31. Alphabetized list of core resources**

Friendly Name (informative)	Resource Type (rt)	Section
Collections	oic.wk.col	D.2
Configuration	oic.wk.con	D.3
Device	oic.wk.d	D.4
Discoverable Resources	oic.wk.res	D.8
Maintenance	oic.wk.mnt	D.5
Platform	oic.wk.p	D.6
Ping	oic.wk.ping	D.7
Resource Directory	oic.wk.rd	D.12
Scenes (Top Level)	oic.wk.sceneList	D.9
Scenes Collections	oic.wk.sceneCollection	D.10
Scenes Member	oic.wk.sceneMember	D.11

3050

3051 D.2 OCF Collection

3052 D.2.1 Introduction

3053 OCF Collection Resource Type contains properties and links. The oic.if.baseline interface exposes
3054 a representation of the links and the properties of the collection resource itself

3055 D.2.2 Fixed URI

3056 /CollectionBaselineInterfaceURI

3057 D.2.3 Resource Type

3058 The resource type (rt) is defined as: oic.wk.col.

3059 D.2.4 RAML Definition

```
3060 #%RAML 0.8
3061 title: Collections
3062 version: 1.0
3063 traits:
3064   - interface-ll :
3065     queryParameters:
3066       if:
3067         enum: ["oic.if.ll"]
3068   - interface-b :
3069     queryParameters:
3070       if:
3071         enum: ["oic.if.b"]
3072   - interface-baseline :
3073     queryParameters:
3074       if:
3075         enum: ["oic.if.baseline"]
3076
3077 /CollectionBaselineInterfaceURI:
3078   description: |
3079     OCF Collection Resource Type contains properties and links.
3080     The oic.if.baseline interface exposes a representation of
3081     the links and the properties of the collection resource itself
3082
3083   is : ['interface-baseline']
3084   get:
3085     description: |
3086       Retrieve on Baseline Interface
3087
3088     responses :
3089       200:
3090         body:
3091           application/json:
3092             schema: /
3093               {
3094                 "$schema": "http://json-schema.org/draft-04/schema#",
3095                 "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3096 reserved.",
3097                 "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3098 schema.json#",
3099                 "title": "Collection",
3100                 "definitions": {
3101                   "uuid": {
3102                     "type": "string",
3103                     "pattern": "[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
3104 [a-fA-F0-9]{12}$"
3105                   },
3106                   "oic.collection.setoflinks": {
3107                     "description": "A set (array) of simple or individual OIC Links. In
3108 addition to properties required for an OIC Link, the identifier for that link in this set is also
3109 required",
3110                     "type": "array",
3111                     "items": {
3112                       "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3113                     }
3114                   },
3115                   "oic.collection.tags": {
3116                     "type": "object",
3117                     "description": "The tags that can be used for tagging links in a
3118 collection",
```

```

3119         "properties": {
3120             "n": {
3121                 "type": "string",
3122                 "description": "Used to name i.e. tag the set of links"
3123             },
3124             "id": {
3125                 "description": "Id for each set of links i.e. tag. Can be an
3126 value that is unique to the use context or a UUIDv4",
3127                 "anyOf": [
3128                     {
3129                         "type": "integer",
3130                         "description": "A number that is unique to that
3131 collection; like an ordinal number that is not repeated"
3132                     },
3133                     {
3134                         "type": "string",
3135                         "description": "A unique string that could be a hash or
3136 similarly unique"
3137                     },
3138                     {
3139                         "$ref": "#/definitions/uuid",
3140                         "description": "A unique string that could be a UUIDv4"
3141                     }
3142                 ]
3143             },
3144             "di": {
3145                 "$ref": "#/definitions/uuid",
3146                 "description": "The device ID which is an UUIDv4 string"
3147             },
3148             "base": {
3149                 "type": "string",
3150                 "description": "The base URI to be used if the links are relative
3151 URIs (i.e. relative references); see base URI in Core spec for details",
3152                 "format": "uri"
3153             }
3154         },
3155         "minProperties": 1
3156     },
3157     "oic.collection.tagged-setoflinks": {
3158         "type": "array",
3159         "description": "A tagged link is a set (array) of links that are tagged
3160 with one or more key-value pairs usually either an ID or Name or both",
3161         "items": [
3162             {
3163                 "$ref": "#/definitions/oic.collection.tags"
3164             },
3165             {
3166                 "$ref": "#/definitions/oic.collection.setoflinks"
3167             }
3168         ],
3169         "additionalItems": false
3170     },
3171     "oic.collection.setof-tagged-setoflinks": {
3172         "type": "array",
3173         "items": [
3174             {
3175                 "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3176             }
3177         ],
3178         "additionalItems": false
3179     },
3180     "oic.collection.alllinks": {
3181         "description": "All forms of links in a collection",
3182         "oneOf": [
3183             {
3184                 "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
3185             },
3186             {
3187                 "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3188             },
3189             {

```

```

3190         "$ref": "#/definitions/oic.collection.setoflinks"
3191     }
3192 ]
3193 },
3194 "oic.collection": {
3195     "type": "object",
3196     "description": "A collection is a set (array) of tagged-link or set
3197 (array) of simple links along with additional properties to describe the collection itself",
3198     "properties": {
3199         "n": {
3200             "type": "string",
3201             "description": "User friendly name of the
3202 collection"
3203         },
3204         "id": {
3205             "anyOf": [
3206                 {
3207                     "type": "integer",
3208                     "description": "A number that is unique to that
3209 collection; like an ordinal number that is not repeated"
3210                 },
3211                 {
3212                     "type": "string",
3213                     "description": "A unique string that could be a hash or
3214 similarly unique"
3215                 }
3216             ],
3217             "$ref": "#/definitions/uuid",
3218             "description": "A unique string that could be a UUIDv4"
3219         },
3220         "description": "ID for the collection. Can be an value that is
3221 unique to the use context or a UUIDv4"
3222     },
3223     "di": {
3224         "$ref": "#/definitions/uuid",
3225         "description": "The device ID which is an UUIDv4 string; used for
3226 backward compatibility with Spec A defintion of /oic/res"
3227     },
3228     "rts": {
3229         "type": "string",
3230         "description": "Defines the list of allowable resource types (for
3231 Target and anchors) in links included in the collection; new links being created can only be from
3232 this list"
3233     },
3234     "drel": {
3235         "type": "string",
3236         "description": "When specified this is the default relationship
3237 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3238     },
3239     "links": {
3240         "$ref": "#/definitions/oic.collection.alllinks"
3241     }
3242 }
3243 },
3244 "type": "object",
3245 "allof": [
3246     {
3247         "$ref": "#/definitions/oic.collection"
3248     }
3249 ]
3250 }
3251
3252 example: /
3253 {
3254     "rt": ["oic.wk.col"],
3255     "id": "unique_example_id",
3256     "rts": [ "oic.r.switch.binary", "oic.r.airFlow" ],
3257     "links": [
3258         {
3259             "href": "switch",

```

```

3260         "rt": "oic.r.switch.binary",
3261         "if": "oic.if.a"
3262     },
3263     {
3264         "href": "airFlow",
3265         "rt": "oic.r.airFlow",
3266         "if": "oic.if.a"
3267     }
3268 ]
3269 }
3270
3271 post:
3272     description: |
3273         Update on Baseline Interface
3274
3275     body:
3276         application/json:
3277             schema: /
3278                 {
3279                     "$schema": "http://json-schema.org/draft-04/schema#",
3280                     "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3281 reserved.",
3282                     "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3283 schema.json#",
3284                     "title": "Collection",
3285                     "definitions": {
3286                         "uuid": {
3287                             "type": "string",
3288                             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
3289 fA-F0-9]{12}$"
3290                         },
3291                         "oic.collection.setoflinks": {
3292                             "description": "A set (array) of simple or individual OIC Links. In addition
3293 to properties required for an OIC Link, the identifier for that link in this set is also required",
3294                             "type": "array",
3295                             "items": {
3296                                 "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3297                             }
3298                         },
3299                         "oic.collection.tags": {
3300                             "type": "object",
3301                             "description": "The tags that can be used for tagging links in a collection",
3302                             "properties": {
3303                                 "n": {
3304                                     "type": "string",
3305                                     "description": "Used to name i.e. tag the set of links"
3306                                 },
3307                                 "id": {
3308                                     "description": "Id for each set of links i.e. tag. Can be an value
3309 that is unique to the use context or a UUIDv4",
3310                                     "anyOf": [
3311                                         {
3312                                             "type": "integer",
3313                                             "description": "A number that is unique to that collection;
3314 like an ordinal number that is not repeated"
3315                                         },
3316                                         {
3317                                             "type": "string",
3318                                             "description": "A unique string that could be a hash or
3319 similarly unique"
3320                                         },
3321                                         {
3322                                             "$ref": "#/definitions/uuid",
3323                                             "description": "A unique string that could be a UUIDv4"
3324                                         }
3325                                     ]
3326                                 },
3327                                 "di": {

```



```

3328         "$ref": "#/definitions/uuid",
3329         "description": "The device ID which is an UUIDv4 string"
3330     },
3331     "base": {
3332         "type": "string",
3333         "description": "The base URI to be used if the links are relative
3334 URIs (i.e. relative references); see base URI in Core spec for details",
3335         "format": "uri"
3336     }
3337 },
3338     "minProperties": 1
3339 },
3340 "oic.collection.tagged-setoflinks": {
3341     "type": "array",
3342     "description": "A tagged link is a set (array) of links that are tagged with
3343 one or more key-value pairs usually either an ID or Name or both",
3344     "items": [
3345         {
3346             "$ref": "#/definitions/oic.collection.tags"
3347         },
3348         {
3349             "$ref": "#/definitions/oic.collection.setoflinks"
3350         }
3351     ],
3352     "additionalItems": false
3353 },
3354 "oic.collection.setof-tagged-setoflinks": {
3355     "type": "array",
3356     "items": [
3357         {
3358             "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3359         }
3360     ],
3361     "additionalItems": false
3362 },
3363 "oic.collection.alllinks": {
3364     "description": "All forms of links in a collection",
3365     "oneOf": [
3366         {
3367             "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
3368         },
3369         {
3370             "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3371         },
3372         {
3373             "$ref": "#/definitions/oic.collection.setoflinks"
3374         }
3375     ]
3376 },
3377 "oic.collection": {
3378     "type": "object",
3379     "description": "A collection is a set (array) of tagged-link or set (array)
3380 of simple links along with additional properties to describe the collection itself",
3381     "properties": {
3382         "n": {
3383             "type": "string",
3384             "description": "User friendly name of the
3385 collection"
3386         },
3387         "id": {
3388             "anyOf": [
3389                 {
3390                     "type": "integer",
3391                     "description": "A number that is unique to that collection;
3392 like an ordinal number that is not repeated"
3393                 },
3394                 {
3395                     "type": "string",
3396                     "description": "A unique string that could be a hash or
3397 similarly unique"
3398                 }
3399             ],
3400         }
3401     }
3402 }

```

```

3399         "$ref": "#/definitions/uuid",
3400         "description": "A unique string that could be a UUIDv4"
3401     },
3402     ],
3403     "description": "ID for the collection. Can be an value that is unique
3404 to the use context or a UUIDv4"
3405 },
3406 "di": {
3407     "$ref": "#/definitions/uuid",
3408     "description": "The device ID which is an UUIDv4 string; used for
3409 backward compatibility with Spec A defintion of /oic/res"
3410 },
3411 "rts": {
3412     "type": "string",
3413     "description": "Defines the list of allowable resource types (for
3414 Target and anchors) in links included in the collection; new links being created can only be from
3415 this list"
3416 },
3417 "drel": {
3418     "type": "string",
3419     "description": "When specified this is the default relationship to
3419 use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3420 },
3421 "links": {
3422     "$ref": "#/definitions/oic.collection.alllinks"
3423 }
3424 }
3425 }
3426 },
3427 "type": "object",
3428 "allOf": [
3429     {
3430         "$ref": "#/definitions/oic.collection"
3431     }
3432 ]
3433 }
3434
3435 responses :
3436     200:
3437         body:
3438             application/json:
3439                 schema: /
3440                 {
3441                     "$schema": "http://json-schema.org/draft-04/schema#",
3442                     "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3443 reserved.",
3444                     "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3445 schema.json#",
3446                     "title": "Collection",
3447                     "definitions": {
3448                         "uuid": {
3449                             "type": "string",
3450                             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
3451 [a-fA-F0-9]{12}$"
3452                         },
3453                         "oic.collection.setoflinks": {
3454                             "description": "A set (array) of simple or individual OIC Links. In
3455 addition to properties required for an OIC Link, the identifier for that link in this set is also
3456 required",
3457                             "type": "array",
3458                             "items": {
3459                                 "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3460                             }
3461                         },
3462                         "oic.collection.tags": {
3463                             "type": "object",
3464                             "description": "The tags that can be used for tagging links in a
3465 collection",
3466                             "properties": {

```

```

3467         "n": {
3468             "type": "string",
3469             "description": "Used to name i.e. tag the set of links"
3470         },
3471         "id": {
3472             "description": "Id for each set of links i.e. tag. Can be an
3473 value that is unique to the use context or a UUIDv4",
3474             "anyOf": [
3475                 {
3476                     "type": "integer",
3477                     "description": "A number that is unique to that
3478 collection; like an ordinal number that is not repeated"
3479                 },
3480                 {
3481                     "type": "string",
3482                     "description": "A unique string that could be a hash or
3483 similarly unique"
3484                 },
3485                 {
3486                     "$ref": "#/definitions/uuid",
3487                     "description": "A unique string that could be a UUIDv4"
3488                 }
3489             ]
3490         },
3491         "di": {
3492             "$ref": "#/definitions/uuid",
3493             "description": "The device ID which is an UUIDv4 string"
3494         },
3495         "base": {
3496             "type": "string",
3497             "description": "The base URI to be used if the links are relative
3498 URIs (i.e. relative references); see base URI in Core spec for details",
3499             "format": "uri"
3500         }
3501     },
3502     "minProperties": 1
3503 },
3504 "oic.collection.tagged-setoflinks": {
3505     "type": "array",
3506     "description": "A tagged link is a set (array) of links that are tagged
3507 with one or more key-value pairs usually either an ID or Name or both",
3508     "items": [
3509         {
3510             "$ref": "#/definitions/oic.collection.tags"
3511         },
3512         {
3513             "$ref": "#/definitions/oic.collection.setoflinks"
3514         }
3515     ],
3516     "additionalItems": false
3517 },
3518 "oic.collection.setof-tagged-setoflinks": {
3519     "type": "array",
3520     "items": [
3521         {
3522             "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3523         }
3524     ],
3525     "additionalItems": false
3526 },
3527 "oic.collection.alllinks": {
3528     "description": "All forms of links in a collection",
3529     "oneOf": [
3530         {
3531             "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
3532         },
3533         {
3534             "$ref": "#/definitions/oic.collection.tagged-setoflinks"
3535         },
3536         {
3537             "$ref": "#/definitions/oic.collection.setoflinks"

```

```

3538         }
3539     ]
3540 },
3541 "oic.collection": {
3542     "type": "object",
3543     "description": "A collection is a set (array) of tagged-link or set
3544 (array) of simple links along with additional properties to describe the collection itself",
3545     "properties": {
3546         "n": {
3547             "type": "string",
3548             "description": "User friendly name of the
3549 collection"
3550         },
3551         "id": {
3552             "anyOf": [
3553                 {
3554                     "type": "integer",
3555                     "description": "A number that is unique to that
3556 collection; like an ordinal number that is not repeated"
3557                 },
3558                 {
3559                     "type": "string",
3560                     "description": "A unique string that could be a hash or
3561 similarly unique"
3562                 },
3563                 {
3564                     "$ref": "#/definitions/uuid",
3565                     "description": "A unique string that could be a UUIDv4"
3566                 }
3567             ],
3568             "description": "ID for the collection. Can be a value that is
3569 unique to the use context or a UUIDv4"
3570         },
3571         "di": {
3572             "$ref": "#/definitions/uuid",
3573             "description": "The device ID which is an UUIDv4 string; used for
3574 backward compatibility with Spec A definition of /oic/res"
3575         },
3576         "rts": {
3577             "type": "string",
3578             "description": "Defines the list of allowable resource types (for
3579 Target and anchors) in links included in the collection; new links being created can only be from
3580 this list"
3581         },
3582         "drel": {
3583             "type": "string",
3584             "description": "When specified this is the default relationship
3585 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3586         },
3587         "links": {
3588             "$ref": "#/definitions/oic.collection.alllinks"
3589         }
3590     }
3591 },
3592 "type": "object",
3593 "allof": [
3594     {
3595         "$ref": "#/definitions/oic.collection"
3596     }
3597 ]
3598 }

```

3599 D.2.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
href	string	yes	Read Write	This is the target URI, it can be specified as a Relative

				Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique.
rel	string		Read Write	
rt	array	yes	Read Write	
if	array	yes	Read Write	
di	string		Read Write	The Device ID on which the Relative Reference in href is to be resolved on. Base URI should be used in preference where possible
huri	string		Read Write	The base URI used to fully qualify a Relative Reference in the href parameter. Use the OCF Schema for URI
p			Read Write	Specifies the framework policies on the Resource referenced by the target URI
bm		yes	Read Write	Specifies the framework policies on the Resource referenced by the target URI for e.g. observable and discoverable
sec			Read Write	Specifies if security needs to be turned on when looking to interact with the Resource
port			Read Write	Secure port to be used for connection
bp	string		Read Write	Batch Parameters: Uri Parameters To Use With An Oic.If.B Batch

				Request Using This Link
anchor	string		Read Write	This is used to override the context URI e.g. override the URI of the containing collection
ins	object		Read Write	

3600 **D.2.6 CRUDN Behaviour**

Resource	Create	Read	Update	Delete	Notify
/CollectionBaselineInterfaceURI		get	post		

3601 **D.2.7 Referenced JSON schemas**

3602 **D.2.8 oic.oic-link-schema.json**

```

3603 {
3604   "$schema": "http://json-schema.org/draft-04/schema#",
3605   "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
3606   "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
3607   "definitions": {
3608     "oic.oic-link": {
3609       "type": "object",
3610       "properties": {
3611         "href": {
3612           "type": "string",
3613           "maxLength": 256,
3614           "description": "This is the target URI, it can be specified as a Relative Reference or
3615 fully-qualified URI. Relative Reference should be used along with the di parameter to make it
3616 unique.",
3617           "format": "uri"
3618         },
3619         "rel": {
3620           "type": "string",
3621           "default": "hosts",
3622           "maxLength": 64,
3623           "description": "The relation of the target URI referenced by the link to the context URI"
3624         },
3625         "rt": {
3626           "type": "array",
3627           "items": [
3628             {
3629               "type": "string",
3630               "maxLength": 64
3631             }
3632           ],
3633           "minItems": 1,
3634           "readOnly": true,
3635           "description": "Resource Type"
3636         },
3637         "if": {
3638           "type": "array",
3639           "items": [
3640             {
3641               "type": "string",
3642               "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
3643 "oic.if.a", "oic.if.s"]
3644             }
3645           ],
3646           "minItems": 1,
3647           "readOnly": true,
3648           "description": "The interface set supported by this resource"
3649         },
3650         "di": {
3651           "type": "string",
3652           "description": "The Device ID on which the Relative Reference in href is to be resolved

```

```

3653 on. Base URI should be used in preference where possible",
3654     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
3655 9]{12}$"
3656   },
3657   "buri": {
3658     "type": "string",
3659     "description": "The base URI used to fully qualify a Relative Reference in the href
3660 parameter. Use the OCF Schema for URI",
3661     "maxLength": 256,
3662     "format": "uri"
3663   },
3664   "p": {
3665     "readOnly": true,
3666     "description": "Specifies the framework policies on the Resource referenced by the target
3667 URI",
3668     "type": "object",
3669     "properties": {
3670       "bm": {
3671         "readOnly": true,
3672         "description": "Specifies the framework policies on the Resource referenced by the
3673 target URI for e.g. observable and discoverable",
3674         "type": "integer"
3675       },
3676       "sec": {
3677         "readOnly": true,
3678         "description": "Specifies if security needs to be turned on when looking to interact
3679 with the Resource",
3680         "type": "boolean"
3681       },
3682       "port": {
3683         "readOnly": true,
3684         "description": "Secure port to be used for connection",
3685         "type": "integer"
3686       }
3687     },
3688     "required" : ["bm"]
3689   },
3690   "bp": {
3691     "type": "string",
3692     "description": " Batch Parameters: URI parameters to use with an oic.if.b batch request
3693 using this link"
3694   },
3695   "title": {
3696     "type": "string",
3697     "maxLength": 64,
3698     "description": "A title for the link relation. Can be used by the UI to provide a
3699 context"
3700   },
3701   "anchor": {
3702     "type": "string",
3703     "maxLength": 256,
3704     "description": "This is used to override the context URI e.g. override the URI of the
3705 containing collection",
3706     "format": "uri"
3707   },
3708   "ins": {
3709     "oneOf": [
3710       {
3711         "type": "integer",
3712         "description": "An ordinal number that is not repeated - must be unique in the
3713 collection context"
3714       },
3715       {
3716         "type": "string",
3717         "maxLength": 256,
3718         "format": "uri",
3719         "description": "Any unique string including a URI"
3720       }
3721     ],
3722     "type": "string",
3723     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-

```

```

3724 9]{12}$",
3725         "description": "Use UUID for universal uniqueness - used in /oic/res to identify the
3726 device"
3727     }
3728     ],
3729     "description": "The instance identifier for this web link in an array of web links - used
3730 in collections"
3731     },
3732     "type": {
3733         "type": "array",
3734         "description": "A hint at the representation of the resource referenced by the target
3735 URI. This represents the media types that are used for both accepting and emitting",
3736         "items": [
3737             {
3738                 "type": "string",
3739                 "maxLength": 64
3740             }
3741         ],
3742         "minItems": 1,
3743         "default": "application/cbor"
3744     }
3745     },
3746     "required": [ "href", "rt", "if" ]
3747 }
3748 },
3749 "type": "object",
3750 "allof": [
3751     { "$ref": "#/definitions/oic.oic-link" }
3752 ]
3753 }
3754

```

3755 **D.3 OIC Configuration**

3756 **D.3.1 Introduction**

3757 Known resource that is hosted by every Server. Allows for device specific information to be
3758 configured.

3759 **D.3.2 Fixed URI**

3760 /oic/con

3761 **D.3.3 Resource Type**

3762 The resource type (rt) is defined as: oic.wk.con.

3763 **D.3.4 RAML Definition**

```

3764 #%RAML 0.8
3765 title: OIC Configuration
3766 version: v1-20160622
3767 traits:
3768   - interface :
3769       queryParameters:
3770           if:
3771               enum: ["oic.if.rw", "oic.if.baseline"]
3772
3773 /oic/con:
3774     description: |
3775       Known resource that is hosted by every Server.
3776       Allows for device specific information to be configured.
3777
3778     is : ['interface']
3779     get:
3780         description: |

```



```

3781         Retrieves the current configuration settings
3782
3783     responses :
3784         200:
3785             body:
3786                 application/json:
3787                 schema: /
3788                     {
3789                         "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-
3790 schema.json#",
3791                         "$schema": "http://json-schema.org/draft-04/schema#",
3792                         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3793 reserved.",
3794                         "definitions": {
3795                             "oic.wk.con": {
3796                                 "type": "object",
3797                                 "properties": {
3798                                     "n": {
3799                                         "type": "string",
3800                                         "maxLength": 64,
3801                                         "description": "Human friendly name"
3802                                     },
3803                                     "loc": {
3804                                         "type": "string",
3805                                         "description": "Location information"
3806                                     },
3807                                     "locn": {
3808                                         "type": "string",
3809                                         "maxLength": 64,
3810                                         "description": "Human Friendly Name"
3811                                     },
3812                                     "c": {
3813                                         "type": "string",
3814                                         "maxLength": 64,
3815                                         "description": "Currency"
3816                                     },
3817                                     "r": {
3818                                         "type": "string",
3819                                         "maxLength": 64,
3820                                         "description": "Region"
3821                                     }
3822                                 }
3823                             },
3824                             "type": "object",
3825                             "allOf": [
3826                                 { "$ref": "#/definitions/oic.wk.con" }
3827                             ],
3828                             "required": [ "n" ]
3829                         }
3830
3831                 example: /
3832                     {
3833                         "rt": ["oic.wk.con"],
3834                         "n": "My Friendly Device Name",
3835                         "loc": "My Location Information",
3836                         "locn": "My Location Name",
3837                         "c": "USD",
3838                         "r": "MyRegion"
3839                     }
3840
3841     post:
3842         description: |
3843             Update the information about the Device
3844
3845

```

```

3846     body:
3847     application/json:
3848         schema: /
3849         {
3850             "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-schema.json#",
3851             "$schema": "http://json-schema.org/draft-04/schema#",
3852             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3853 reserved.",
3854             "definitions": {
3855                 "oic.wk.con": {
3856                     "type": "object",
3857                     "properties": {
3858                         "n": {
3859                             "type": "string",
3860                             "maxLength": 64,
3861                             "description": "Human friendly name"
3862                         },
3863                         "loc": {
3864                             "type": "string",
3865                             "description": "Location information"
3866                         },
3867                         "locn": {
3868                             "type": "string",
3869                             "maxLength": 64,
3870                             "description": "Human Friendly Name"
3871                         },
3872                         "c": {
3873                             "type": "string",
3874                             "maxLength": 64,
3875                             "description": "Currency"
3876                         },
3877                         "r": {
3878                             "type": "string",
3879                             "maxLength": 64,
3880                             "description": "Region"
3881                         }
3882                     }
3883                 },
3884             },
3885             "type": "object",
3886             "allOf": [
3887                 { "$ref": "#/definitions/oic.wk.con" }
3888             ],
3889             "required": [ "n" ]
3890         }
3891
3892     example: /
3893     {
3894         "n": "My Friendly Device Name"
3895     }
3896
3897     responses :
3898     200:
3899     body:
3900     application/json:
3901     schema: /
3902     {
3903         "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-
3904 schema.json#",
3905         "$schema": "http://json-schema.org/draft-04/schema#",
3906         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3907 reserved.",
3908         "definitions": {
3909             "oic.wk.con": {
3910                 "type": "object",
3911                 "properties": {

```

```

3912         "n": {
3913             "type": "string",
3914             "maxLength": 64,
3915             "description": "Human friendly name"
3916         },
3917         "loc": {
3918             "type": "string",
3919             "description": "Location information"
3920         },
3921         "locn": {
3922             "type": "string",
3923             "maxLength": 64,
3924             "description": "Human Friendly Name"
3925         },
3926         "c": {
3927             "type": "string",
3928             "maxLength": 64,
3929             "description": "Currency"
3930         },
3931         "r": {
3932             "type": "string",
3933             "maxLength": 64,
3934             "description": "Region"
3935         }
3936     }
3937 },
3938 },
3939 "type": "object",
3940 "allOf": [
3941     { "$ref": "#/definitions/oic.wk.con" }
3942 ],
3943 "required": [ "n" ]
3944 }
3945
3946 example: /
3947 {
3948     "n": "My Friendly Device Name"
3949 }
3950

```

3951 D.3.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.
n	string	yes	Read Write	Human friendly name
loc	string		Read Write	Location information
locn	string		Read Write	Human Friendly Name
c	string		Read Write	Currency
r	string		Read Write	Region

3952 D.3.6 CRUDN Behaviour

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

3953 D.4 Device

3954 D.4.1 Introduction

3955 Known resource that is hosted by every Server. Allows for logical device specific information to be
3956 discovered.

3957 D.4.2 Fixed URI

3958 /oic/d

3959 D.4.3 Resource Type

3960 The resource type (rt) is defined as: oic.wk.d.

3961 D.4.4 RAML Definition

3962 `##RAML 0.8`

3963 `title: OIC Root Device`

3964 `version: v1-20160622`

3965 `traits:`

3966 `- interface :`

3967 `queryParameters:`

3968 `if:`

3969 `enum: ["oic.if.r", "oic.if.baseline"]`

3970

3971 `/oic/d:`

3972 `description: |`

3973 `Known resource that is hosted by every Server.`

3974 `Allows for logical device specific information to be discovered.`

3975

3976 `is : ['interface']`

3977 `get:`

3978 `description: |`

3979 `Retrieve the information about the Device`

3980

3981 `responses :`

3982 `200:`

3983 `body:`

3984 `application/json:`

3985 `schema: /`

3986 `{`

3987 `"$schema": "http://json-schemas.org/draft-04/schema#",`

3988 `"description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights`

3989 `reserved.",`

3990 `"id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.d-`

3991 `schema.json#",`

3992 `"definitions": {`

3993 `"uuid": {`

3994 `"type": "string",`

3995 `"pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-`

3996 `fA-F0-9]{12}$"`

3997 `},`

3998 `"oic.wk.d": {`

3999 `"type": "object",`

4000 `"properties": {`

4001 `"n": {`

4002 `"type": "string",`

4003 `"maxLength": 64,`

4004 `"readOnly": true,`

4005 `"description": "Human friendly name"`

4006 `},`

4007 `"di": {`

```

4008         "$ref": "#/definitions/uuid",
4009         "readOnly": true,
4010         "description": "Unique identifier for device (UUID)"
4011     },
4012     "icv": {
4013         "type": "string",
4014         "maxLength": 64,
4015         "readOnly": true,
4016         "description": "The version of the OIC Server"
4017     },
4018     "dmv": {
4019         "type": "string",
4020         "maxLength": 64,
4021         "readOnly": true,
4022         "description": "The spec version of the vertical and/or resource
4023 specification"
4024     }
4025 }
4026 }
4027 },
4028 "type": "object",
4029 "allOf": [
4030     { "$ref": "#/definitions/oic.wk.d" }
4031 ],
4032 "required": [ "n", "di", "icv", "dmv" ]
4033 }
4034
4035 example: /
4036 {
4037     "n": "Device 1",
4038     "rt": [ "oic.wk.d" ],
4039     "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
4040     "icv": "core.1.1.0",
4041     "dmv": "res.1.1.0"
4042 }
4043

```

4044 D.4.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string	yes	Read Only	
di		yes	Read Only	Unique identifier for device (UUID)
icv	string	yes	Read Only	
dmv	string	yes	Read Only	

4045 D.4.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/d		get			

4046 D.5 Maintenance

4047 D.5.1 Introduction

4048 The resource through which an Device is maintained and can be used for diagnostic purposes. fr
4049 (Factory Reset) is a boolean. The value 0 means No action (Default), the value 1 means Start
4050 Factory Reset After factory reset, this value shall be changed back to the default value rb (Reboot)
4051 is a boolean. The value 0 means No action (Default), the value 1 means Start Reboot After Reboot,
4052 this value shall be changed back to the default value

4053 D.5.2 Fixed URI

4054 /oic/mt

4055 D.5.3 Resource Type

4056 The resource type (rt) is defined as: oic.wk.mnt.

4057 D.5.4 RAML Definition

```
4058 #%RAML 0.8
4059 title: Maintenance
4060 version: v1-20160622
4061 traits:
4062   - interface :
4063     queryParameters:
4064       if:
4065         enum: ["oic.if.r", "oic.if.baseline"]
4066
4067 /oic/mnt:
4068   description: |
4069     The resource through which an Device is maintained and can be used for diagnostic purposes.
4070     fr (Factory Reset) is a boolean.
4071     The value 0 means No action (Default), the value 1 means Start Factory Reset
4072     After factory reset, this value shall be changed back to the default value
4073     rb (Reboot) is a boolean.
4074     The value 0 means No action (Default), the value 1 means Start Reboot
4075     After Reboot, this value shall be changed back to the default value
4076
4077   is : ['interface']
4078   get:
4079     description: |
4080       Retrieve the maintenance action status
4081
4082     queryParameters:
4083       if:
4084         enum: oic.if.r
4085     responses :
4086       200:
4087         body:
4088           application/json:
4089             schema: /
4090             {
4091               "$schema": "http://json-schemas.org/draft-04/schema#",
4092               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4093 reserved.",
4094               "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
4095 schema.json#",
4096               "definitions": {
4097                 "oic.wk.mnt": {
4098                   "type": "object",
4099                   "properties": {
4100                     "n": {
4101                       "type" : "string",
4102                       "maxLength" : 64,
4103                       "description": "Name"
4104                     },
4105                     "fr":{
4106                       "type": "boolean",
4107                       "description": "Factory Reset"
4108                     },
4109                     "rb": {
4110                       "type": "boolean",
4111                       "description": "Reboot Action"
4112                     }
4113                   }
4114                 }
4115             }
```

```

4114     },
4115     },
4116     "type": "object",
4117     "allOf": [
4118     { "$ref": "#/definitions/oic.wk.mnt" }
4119     ],
4120     "required": ["fr"]
4121 }
4122
4123 example: /
4124 {
4125     "rt":  ["oic.wk.mnt"],
4126     "n":   "My Maintenance Actions",
4127     "fr":  false,
4128     "rb":  false
4129 }
4130
4131 post:
4132 description: |
4133     Set the maintenance action(s)
4134
4135 queryParameters:
4136     if:
4137         enum: oic.if.rw
4138 body:
4139     application/json:
4140         schema: /
4141             {
4142             "$schema": "http://json-schemas.org/draft-04/schema#",
4143             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4144 reserved.",
4145             "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-schema.json#",
4146             "definitions": {
4147             "oic.wk.mnt": {
4148             "type": "object",
4149             "properties": {
4150             "n": {
4151             "type" : "string",
4152             "maxLength" : 64,
4153             "description": "Name"
4154             },
4155             "fr":{
4156             "type": "boolean",
4157             "description": "Factory Reset"
4158             },
4159             "rb": {
4160             "type": "boolean",
4161             "description": "Reboot Action"
4162             }
4163             }
4164             }
4165             },
4166             "type": "object",
4167             "allOf": [
4168             { "$ref": "#/definitions/oic.wk.mnt" }
4169             ],
4170             "required": ["fr"]
4171             }
4172
4173 example: /
4174 {
4175     "n":   "My Maintenance Actions",
4176     "fr":  false,
4177     "rb":  false

```

```

4178         }
4179
4180     responses :
4181         200:
4182             body:
4183                 application/json:
4184                     schema: /
4185                         {
4186                             "$schema": "http://json-schemas.org/draft-04/schema#",
4187                             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4188 reserved.",
4189                             "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
4190 schema.json#",
4191                             "definitions": {
4192                                 "oic.wk.mnt": {
4193                                     "type": "object",
4194                                     "properties": {
4195                                         "n": {
4196                                             "type" : "string",
4197                                             "maxLength" : 64,
4198                                             "description": "Name"
4199                                         },
4200                                         "fr":{
4201                                             "type": "boolean",
4202                                             "description": "Factory Reset"
4203                                         },
4204                                         "rb": {
4205                                             "type": "boolean",
4206                                             "description": "Reboot Action"
4207                                         }
4208                                     }
4209                                 },
4210                             },
4211                             "type": "object",
4212                             "allOf": [
4213                                 { "$ref": "#/definitions/oic.wk.mnt" }
4214                             ],
4215                             "required": ["fr"]
4216                         }
4217
4218                     example: /
4219                         {
4220                             "n":    "My Maintenance Actions",
4221                             "fr":  false,
4222                             "rb":  false
4223                         }
4224

```

4225 D.5.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
n	string		Read Write	Name
fr	boolean	yes	Read Write	Factory Reset
rb	boolean		Read Write	Reboot Action

4226 D.5.6 CRUDN Behaviour

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

4227 D.6 Platform

4228 D.6.1 Introduction

4229 Known resource that is defines the platform on which a Server is hosted. Allows for platform
4230 specific information to be discovered.

4231 D.6.2 Fixed URI

4232 /oic/p

4233 D.6.3 Resource Type

4234 The resource type (rt) is defined as: oic.wk.p.

4235 D.6.4 RAML Definition

4236 `##RAML 0.8`

4237 `title: Platform`

4238 `version: v1-20160622`

4239 `traits:`

4240 `- interface :`

4241 `queryParameters:`

4242 `if:`

4243 `enum: ["oic.if.r", "oic.if.baseline"]`

4244

4245 `/oic/p:`

4246 `description: |`

4247 `Known resource that is defines the platform on which an Server is hosted.`

4248 `Allows for platform specific information to be discovered.`

4249

4250 `is : ['interface']`

4251 `get:`

4252 `description: |`

4253 `Retrieve the information about the Platform`

4254

4255 `responses :`

4256 `200:`

4257 `body:`

4258 `application/json:`

4259 `schema: |`

4260 `{`

4261 `"$schema": "http://json-schemas.org/draft-04/schema#",`

4262 `"description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights`

4263 `reserved.",`

4264 `"id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.p-`

4265 `schema.json#",`

4266 `"definitions": {`

4267 `"uuid": {`

4268 `"type": "string",`

4269 `"pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-`

4270 `fA-F0-9]{12}$"`

4271 `},`

4272 `"oic.wk.p": {`

4273 `"type": "object",`

4274 `"properties": {`

4275 `"pi": {`

4276 `"$ref": "#/definitions/uuid",`

4277 `"readOnly": true,`

4278 `"description": "Platform Identifier as a UUID"`

4279 `},`

4280 `"mnmn": {`

4281 `"type": "string",`

```

4282         "readOnly": true,
4283         "description": "Manufacturer Name",
4284         "maxLength": 64
4285     },
4286     "mnml": {
4287         "type": "string",
4288         "readOnly": true,
4289         "description": "Manufacturer's URL",
4290         "maxLength": 256,
4291         "format": "uri"
4292     },
4293     "mnmo": {
4294         "type": "string",
4295         "maxLength": 64,
4296         "readOnly": true,
4297         "description": "Model number as designated by manufacturer"
4298     },
4299     "mndt": {
4300         "type": "string",
4301         "readOnly": true,
4302         "description": "Manufacturing Date as defined in ISO 8601, where the format
4303 is [yyyy]-[mm]-[dd].",
4304         "pattern": "^[0-9]{4}-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$"
4305     },
4306     "mnpv": {
4307         "type": "string",
4308         "maxLength": 64,
4309         "readOnly": true,
4310         "description": "Platform Version"
4311     },
4312     "mnos": {
4313         "type": "string",
4314         "maxLength": 64,
4315         "readOnly": true,
4316         "description": "Platform Resident OS Version"
4317     },
4318     "mnhw": {
4319         "type": "string",
4320         "maxLength": 64,
4321         "readOnly": true,
4322         "description": "Platform Hardware Version"
4323     },
4324     "mnfv": {
4325         "type": "string",
4326         "maxLength": 64,
4327         "readOnly": true,
4328         "description": "Manufacturer's firmware version"
4329     },
4330     "mnsl": {
4331         "type": "string",
4332         "readOnly": true,
4333         "description": "Manufacturer's Support Information URL",
4334         "maxLength": 256,
4335         "format": "uri"
4336     },
4337     "st": {
4338         "type": "string",
4339         "readOnly": true,
4340         "description": "Reference time for the device as defined in ISO 8601, where
4341 concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'. The
4342 format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.",
4343         "format": "date-time"
4344     },
4345     "vid": {
4346         "type": "string",
4347         "maxLength": 64,
4348         "readOnly": true,
4349         "description": "Manufacturer's defined string for the platform. The string
4350 is freeform and up to the manufacturer on what text to populate it"
4351     }
4352 }

```

```

4353     }
4354   },
4355   "type": "object",
4356   "allOf": [
4357     { "$ref": "#/definitions/oic.wk.p" }
4358   ],
4359   "required": [ "pi", "mnmn" ]
4360 }
4361

```

```

4362 example: /
4363 {
4364   "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
4365   "rt": [ "oic.wk.p" ],
4366   "mnmn": "Acme, Inc"
4367 }
4368

```

4369 D.6.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
pi		yes	Read Only	Platform Identifier as a UUID
mnmn	string	yes	Read Only	Manufacturer Name
mnml	string		Read Only	Manufacturer's URL
mnmo	string		Read Only	
mndt	string		Read Only	Manufacturing Date as defined in ISO 8601, where the format is [yyyy]-[mm]-[dd].
mpv	string		Read Only	
mnos	string		Read Only	
mnhw	string		Read Only	
mnfv	string		Read Only	
mnsi	string		Read Only	Manufacturer's Support Information URL
st	string		Read Only	Reference time for the device as defined in ISO 8601, where concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.
vid	string		Read Only	

4370 D.6.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/p		get			

4371 D.7 Ping

4372 D.7.1 Introduction

4373 The resource using which an Client keeps its Connection with an Server active.

4374 D.7.2 Fixed URI

4375 /oic/ping

4376 D.7.3 Resource Type

4377 The resource type (rt) is defined as: oic.wk.ping.

4378 D.7.4 RAML Definition

4379 `##RAML 0.8`

4380 `title: Ping`

4381 `version: v1-20160622`

4382 `traits:`

4383 `- interface :`

4384 `queryParameters:`

4385 `if:`

4386 `enum: ["oic.if.rw", "oic.if.baseline"]`

4387

4388 `/oic/ping:`

4389 `description: |`

4390 `The resource using which an Client keeps its Connection with an Server active.`

4391

4392 `is : ['interface']`

4393 `get:`

4394 `description: |`

4395 `Retrieve the ping information`

4396

4397 `responses :`

4398 `200:`

4399 `body:`

4400 `application/json:`

4401 `schema: /`

4402 `{`

4403 `"$schema": "http://json-schemas.org/draft-04/schema#",`

4404 `"description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights`

4405 `reserved.",`

4406 `"id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.ping-`

4407 `schema.json#",`

4408 `"definitions": {`

4409 `"oic.wk.ping": {`

4410 `"type": "object",`

4411 `"properties": {`

4412 `"in": {`

4413 `"type": "integer",`

4414 `"description": "ReadWrite, Indicates the interval for which connection`

4415 `shall be kept alive"`

4416 `}
4417 }
4418 },`

4419 `},`

4420 `"type": "object",`

4421 `"allOf": [`

4422 `{ "$ref": "#/definitions/oic.wk.ping" }`

4423 `],`

4424 `"required": [`

4425 `"in"`

4426 `]`

```

4427     }
4428
4429     example: /
4430     {
4431         "rt": ["oic.wk.ping"],
4432         "n": "Ping Information",
4433         "in": 16
4434     }
4435

```

4436 D.7.5 Property Definition

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

4437 D.7.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/oic/ping		get			

4438 D.8 Discoverable Resources

4439 D.8.1 Introduction

4440 The resource through which the corresponding Server is discovered and introspected for available
4441 resources.

4442 D.8.2 Fixed URI

4443 /oic/res

4444 D.8.3 Resource Type

4445 The resource type (rt) is defined as: oic.wk.res.

4446 D.8.4 RAML Definition

```

4447 #%RAML 0.8
4448 title: Discoverable Resources
4449 version: v1-20160622
4450 traits:
4451   - interface :
4452     queryParameters:
4453       if:
4454         enum: ["oic.if.ll", "oic.if.baseline"]
4455
4456 /oic/res:
4457   description: |
4458     The resource through which the corresponding Server is discovered and introspected for
4459     available resources.
4460
4461   is : ['interface']
4462   get:
4463     description: |
4464       Retrieve the discoverable resource set
4465
4466   responses :

```

```

4467     200:
4468     body:
4469     application/json:
4470     schema: /
4471     {
4472     "$schema": "http://json-schema.org/draft-v4/schema#",
4473     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4474 reserved.",
4475     "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-
4476 schema.json#",
4477     "definitions": {
4478     "uuid": {
4479     "type": "string",
4480     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
4481 fA-F0-9]{12}$"
4482     },
4483     "oic.res-links.json": {
4484     "type": "object",
4485     "properties": {
4486     "n": {
4487     "type": "string",
4488     "maxLength": 64,
4489     "readOnly": true,
4490     "description": "Human friendly name"
4491     },
4492     "di": {
4493     "$ref": "#/definitions/uuid",
4494     "readOnly": true,
4495     "description": "Unique identifier for device (UUID) as indicated by the
4496 /oic/d resource of the device"
4497     },
4498     "mpro": {
4499     "readOnly": true,
4500     "description": "Supported messaging protocols",
4501     "type": "string",
4502     "maxLength": 64
4503     },
4504     "links": {
4505     "type": "array",
4506     "items": {
4507     "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
4508     }
4509     }
4510     },
4511     "required": ["di", "links"]
4512     }
4513     },
4514     "description": "The list of resources expressed as OIC links",
4515     "type": "array",
4516     "items": {
4517     "$ref": "#/definitions/oic.res-links.json"
4518     }
4519     }
4520
4521     example: /
4522     [
4523     {
4524     "rt": ["oic.wk.res"],
4525     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
4526     "links":
4527     [
4528     {
4529     "href": "/res",
4530     "rel": "self",
4531     "rt": ["oic.r.collection"],
4532     "if": ["oic.if.ll"]
4533     },
4534     {

```

```

4535         "href": "/smartDevice",
4536         "rel": "contained",
4537         "rt": ["oic.d.smartDevice"],
4538         "if": ["oic.if.a"]
4539     }
4540 ]
4541 }
4542 ]
4543 ]

```

4544 **D.8.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string		Read Only	
di		yes	Read Only	Unique identifier for device (UUID) as indicated by the /oic/d resource of the device
mpro			Read Write	Supported messaging protocols
links	array	yes	Read Write	

4545 **D.8.6 CRUDN Behaviour**

Resource	Create	Read	Update	Delete	Notify
/oic/res		get			

4546 **D.9 Scenes (Top level)**

4547 **D.9.1 Introduction**

4548 Toplevel Scene resource. This resource is a generic collection resource. The rts value shall contain
4549 oic.sceneCollection resource types.

4550 **D.9.2 Fixed URI**

4551 /SceneListResURI

4552 **D.9.3 Resource Type**

4553 The resource type (rt) is defined as: oic.wk.sceneList.

4554 **D.9.4 RAML Definition**

```

4555 #%RAML 0.8
4556 title: Scene
4557 version: v1-20160622
4558 traits:
4559   - interface :
4560     queryParameters:
4561       if:
4562         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
4563
4564 /SceneListResURI:
4565   description: |
4566     Toplevel Scene resource.
4567     This resource is a generic collection resource.
4568     The rts value shall contain oic.sceneCollection resource types.
4569

```

```

4570 get:
4571     description: |
4572         Provides the current list of web links pointing to scenes
4573
4574     responses :
4575         200:
4576             body:
4577                 application/json:
4578                     schema: /
4579                         {
4580                             "$schema": "http://json-schema.org/draft-04/schema#",
4581                             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4582 reserved.",
4583                             "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
4584 schema.json#",
4585                             "title": "Collection",
4586                             "definitions": {
4587                                 "uuid": {
4588                                     "type": "string",
4589                                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
4590 [a-fA-F0-9]{12}$"
4591                                 },
4592                                 "oic.collection.setoflinks": {
4593                                     "description": "A set (array) of simple or individual OIC Links. In
4594 addition to properties required for an OIC Link, the identifier for that link in this set is also
4595 required",
4596                                     "type": "array",
4597                                     "items": {
4598                                         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
4599                                     }
4600                                 },
4601                                 "oic.collection.tags": {
4602                                     "type": "object",
4603                                     "description": "The tags that can be used for tagging links in a
4604 collection",
4605                                     "properties": {
4606                                         "n": {
4607                                             "type": "string",
4608                                             "description": "Used to name i.e. tag the set of links"
4609                                         },
4610                                         "id": {
4611                                             "description": "Id for each set of links i.e. tag. Can be an
4612 value that is unique to the use context or a UUIDv4",
4613                                             "anyOf": [
4614                                                 {
4615                                                     "type": "integer",
4616                                                     "description": "A number that is unique to that
4617 collection; like an ordinal number that is not repeated"
4618                                                 },
4619                                                 {
4620                                                     "type": "string",
4621                                                     "description": "A unique string that could be a hash or
4622 similarly unique"
4623                                                 },
4624                                                 {
4625                                                     "$ref": "#/definitions/uuid",
4626                                                     "description": "A unique string that could be a UUIDv4"
4627                                                 }
4628                                             ]
4629                                         },
4630                                         "di": {
4631                                             "$ref": "#/definitions/uuid",
4632                                             "description": "The device ID which is an UUIDv4 string"
4633                                         },
4634                                         "base": {
4635                                             "type": "string",
4636                                             "description": "The base URI to be used if the links are relative
4637 URIs (i.e. relative references); see base URI in Core spec for details",

```



```

4638         "format": "uri"
4639     },
4640 },
4641     "minProperties": 1
4642 },
4643     "oic.collection.tagged-setoflinks": {
4644         "type": "array",
4645         "description": "A tagged link is a set (array) of links that are tagged
4646 with one or more key-value pairs usually either an ID or Name or both",
4647         "items": [
4648             {
4649                 "$ref": "#/definitions/oic.collection.tags"
4650             },
4651             {
4652                 "$ref": "#/definitions/oic.collection.setoflinks"
4653             }
4654         ],
4655         "additionalItems": false
4656     },
4657     "oic.collection.setof-tagged-setoflinks": {
4658         "type": "array",
4659         "items": [
4660             {
4661                 "$ref": "#/definitions/oic.collection.tagged-setoflinks"
4662             }
4663         ],
4664         "additionalItems": false
4665     },
4666     "oic.collection.alllinks": {
4667         "description": "All forms of links in a collection",
4668         "oneOf": [
4669             {
4670                 "$ref": "#/definitions/oic.collection.setof-tagged-setoflinks"
4671             },
4672             {
4673                 "$ref": "#/definitions/oic.collection.tagged-setoflinks"
4674             },
4675             {
4676                 "$ref": "#/definitions/oic.collection.setoflinks"
4677             }
4678         ]
4679     },
4680     "oic.collection": {
4681         "type": "object",
4682         "description": "A collection is a set (array) of tagged-link or set
4683 (array) of simple links along with additional properties to describe the collection itself",
4684         "properties": {
4685             "n": {
4686                 "type": "string",
4687                 "description": "User friendly name of the
4688 collection"
4689             },
4690             "id": {
4691                 "anyOf": [
4692                     {
4693                         "type": "integer",
4694                         "description": "A number that is unique to that
4695 collection; like an ordinal number that is not repeated"
4696                     },
4697                     {
4698                         "type": "string",
4699                         "description": "A unique string that could be a hash or
4700 similarly unique"
4701                     },
4702                     {
4703                         "$ref": "#/definitions/uuid",
4704                         "description": "A unique string that could be a UUIDv4"
4705                     }
4706                 ],
4707                 "description": "ID for the collection. Can be an value that is
4708 unique to the use context or a UUIDv4"

```

```

4709         "di": {
4710             "$ref": "#/definitions/uuid",
4711             "description": "The device ID which is an UUIDv4 string; used for
backward compatibility with Spec A definition of /oic/res"
4712         },
4713         "rts": {
4714             "type": "string",
4715             "description": "Defines the list of allowable resource types (for
4716 Target and anchors) in links included in the collection; new links being created can only be from
4717 this list"
4718         },
4719         "drel": {
4720             "type": "string",
4721             "description": "When specified this is the default relationship
4722 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
4723         },
4724         "links": {
4725             "$ref": "#/definitions/oic.collection.alllinks"
4726         }
4727     }
4728 },
4729     "type": "object",
4730     "allOf": [
4731     {
4732         "$ref": "#/definitions/oic.collection"
4733     }
4734     ]
4735 }
4736
4737
4738     example: /
4739     {
4740         "rt": "oic.wk.sceneList",
4741         "n": "list of scene Collections",
4742         "rts": "oic.wk.sceneCollection",
4743         "links": [
4744         ]
4745     }
4746

```

4747 D.9.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
uuid	string		Read Write	
n	string		Read Write	Used to name i.e. tag the set of links
id			Read Write	
di			Read Write	The device ID which is an UUIDv4 string
base	string		Read Write	The base URI to be used if the links are relative URIs (i.e. relative references); see base URI in Core spec for details
n	string		Read Write	User friendly name of the collection
id			Read Write	

di			Read Write	The device ID which is an UUIDv4 string; used for backward compatibility with Spec A definition of /oic/res
rts	string		Read Write	Defines the list of allowable resource types (for Target and anchors) in links included in the collection; new links being created can only be from this list
drel	string		Read Write	When specified this is the default relationship to use when an OIC Link does not specify an explicit relationship with *rel* parameter
links			Read Write	

4748 **D.9.6 CRUDN Behaviour**

Resource	Create	Read	Update	Delete	Notify
/SceneListResURI		get			

4749 **D.10 Scene Collections**

4750 **D.10.1 Introduction**

4751 Collection that models a set of Scenes. This resource is a generic collection resource with
4752 additional parameters. The rts value shall contain oic.sceneMember resource types. The additional
4753 parameters are lastScene, this is the scene value last set by any OIC Client sceneValueList,
4754 this is the list of available scenes lastScene shall be listed in sceneValueList.

4755 **D.10.2 Fixed URI**

4756 /SceneCollectionResURI

4757 **D.10.3 Resource Type**

4758 The resource type (rt) is defined as: oic.wk.sceneCollection.

4759 **D.10.4 RAML Definition**

```

4760 #%RAML 0.8
4761 title: Scene
4762 version: v1-20160622
4763 traits:
4764   - interface :
4765     queryParameters:
4766       if:
4767         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]

```

```

4768
4769 /SceneCollectionResURI:
4770     description: |
4771         Collection that models a set of Scenes.
4772         This resource is a generic collection resource with additional parameters.
4773         The rts value shall contain oic.sceneMember resource types.
4774         The additional parameters are
4775             lastScene, this is the scene value last set by any OIC Client
4776             sceneValueList, this is the list of available scenes
4777             lastScene shall be listed in sceneValueList.
4778
4779     get:
4780         description: |
4781             Provides the current list of web links pointing to scenes
4782
4783     responses :
4784         200:
4785             body:
4786                 application/json:
4787                 schema: /
4788                 {
4789                 "$schema": "http://json-schema.org/draft-04/schema#",
4790                 "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4791 reserved.",
4792                 "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
4793 schema.json#",
4794                 "title" : "Scene Collection",
4795                 "definitions": {
4796                 "oic.sceneCollection": {
4797                 "type": "object",
4798                 "properties": {
4799                 "lastScene": {
4800                 "type": "string",
4801                 "description": "Last selected Scene, shall be part of sceneValues",
4802                 "format": "UTF8"
4803                 },
4804                 "sceneValues": {
4805                 "type": "string",
4806                 "readOnly": true,
4807                 "description": "All available scene values",
4808                 "format": "CSV"
4809                 },
4810                 "n": {
4811                 "type": "string",
4812                 "description": "Used to name the Scene collection",
4813                 "format": "UTF8"
4814                 },
4815                 "id": {
4816                 "type": "string",
4817                 "description" : "A unique string that could be a hash or
4818 similarly unique"
4819                 },
4820                 "rts": {
4821                 "type": "string",
4822                 "readOnly": true,
4823                 "description": "Defines the list of allowable resource types in links
4824 included in the collection; new links being created can only be from this list",
4825                 "format": "UTF8"
4826                 },
4827                 "links": {
4828                 "type": "array",
4829                 "description": "Array of OIC web links that are reference from this
4830 collection",
4831                 "items" : {
4832                 "allOf": [
4833                 { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },

```

```

4834         { "required" : [ "ins" ] }
4835     ]
4836     }
4837 }
4838 },
4839     "required": [ "lastScene","sceneValues","rts","id" ]
4840 }
4841 },
4842
4843     "type": "object",
4844     "allOf" : [
4845     { "$ref": "#/definitions/oic.sceneCollection" }
4846     ]
4847 }
4848
4849     example: /
4850     {
4851         "lastScene": "off",
4852         "sceneValues": "off,Reading,TVWatching",
4853         "rt":         "oic.wk.sceneCollection",
4854         "n":         "My Scenes for my living room",
4855         "id":        "0685B960-736F-46F7-BEC0-9E6CBD671ADCl",
4856         "rts":       "oic.wk.sceneMember",
4857         "links": [
4858             ]
4859     }
4860
4861     put:
4862     description: |
4863         Provides the action to change the last settted scene selection.
4864         Calling this method shall update of all sceneMembers to the prescribed membervalue.
4865         When this method is called with the same value as the current lastScene value
4866         then all sceneMembers shall be updated.
4867
4868     body:
4869     application/json:
4870     schema: /
4871     {
4872         "$schema": "http://json-schema.org/draft-04/schema#",
4873         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4874 reserved.",
4875         "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
4876 schema.json#",
4877         "title" : "Scene Collection",
4878         "definitions": {
4879             "oic.sceneCollection": {
4880                 "type": "object",
4881                 "properties": {
4882                     "lastScene": {
4883                         "type": "string",
4884                         "description": "Last selected Scene, shall be part of sceneValues",
4885                         "format": "UTF8"
4886                     },
4887                     "sceneValues": {
4888                         "type": "string",
4889                         "readOnly": true,
4890                         "description": "All available scene values",
4891                         "format": "CSV"
4892                     },
4893                     "n": {
4894                         "type": "string",
4895                         "description": "Used to name the Scene collection",
4896                         "format": "UTF8"
4897                     },
4898                     "id": {
4899                         "type": "string",
4900                         "description" : "A unique string that could be a hash or

```

```

4901 similarly unique"
4902     },
4903     "rts": {
4904         "type": "string",
4905         "readOnly": true,
4906         "description": "Defines the list of allowable resource types in links included
4907 in the collection; new links being created can only be from this list",
4908         "format": "UTF8"
4909     },
4910     "links": {
4911         "type": "array",
4912         "description": "Array of OIC web links that are reference from this
4913 collection",
4914         "items": {
4915             "allOf": [
4916                 { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
4917                 { "required": [ "ins" ] }
4918             ]
4919         }
4920     },
4921     },
4922     "required": [ "lastScene" ]
4923 }
4924 },
4925
4926 "type": "object",
4927 "allOf" : [
4928     { "$ref": "#/definitions/oic.sceneCollection" }
4929 ]
4930 }
4931
4932 example: /
4933 {
4934     "lastScene": "Reading"
4935 }
4936
4937 responses :
4938 200:
4939     description: |
4940     Indicates that the value is changed.
4941     The changed properties are provided in the response.
4942
4943     body:
4944     application/json:
4945         schema: /
4946         {
4947             "$schema": "http://json-schema.org/draft-04/schema#",
4948             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4949 reserved.",
4950             "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
4951 schema.json#",
4952             "title" : "Scene Collection",
4953             "definitions": {
4954                 "oic.sceneCollection": {
4955                     "type": "object",
4956                     "properties": {
4957                         "lastScene": {
4958                             "type": "string",
4959                             "description": "Last selected Scene, shall be part of sceneValues",
4960                             "format": "UTF8"
4961                         },
4962                         "sceneValues": {
4963                             "type": "string",
4964                             "readOnly": true,
4965                             "description": "All available scene values",
4966                             "format": "CSV"
4967                         }
4968                     }
4969                 }
4970             }
4971         }

```

```

4968         "n": {
4969             "type": "string",
4970             "description": "Used to name the Scene collection",
4971             "format": "UTF8"
4972         },
4973         "id": {
4974             "type": "string",
4975             "description" : "A unique string that could be a hash or
similarly unique"
4976         },
4977     },
4978     "rts": {
4979         "type": "string",
4980         "readOnly": true,
4981         "description": "Defines the list of allowable resource types in links
included in the collection; new links being created can only be from this list",
4982         "format": "UTF8"
4983     },
4984     },
4985     "links": {
4986         "type": "array",
4987         "description": "Array of OIC web links that are reference from this
collection",
4988     },
4989     "items" : {
4990         "allOf": [
4991             { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
4992             { "required" : [ "ins" ] }
4993         ]
4994     }
4995 },
4996 },
4997 "required": [ "lastScene" ]
4998 }
4999 },
5000
5001 "type": "object",
5002 "allof" : [
5003     { "$ref": "#/definitions/oic.sceneCollection" }
5004 ]
5005 }
5006
5007 example: /
5008 {
5009     "lastScene": "Reading"
5010 }
5011

```

5012 D.10.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id		yes	Read Write	
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

5013 **D.10.6 CRUDN Behaviour**

Resource	Create	Read	Update	Delete	Notify
/SceneCollectionResURI	put	get			

5014 **D.11 Scene Member**

5015 **D.11.1 Introduction**

5016 Collection that models a sceneMember.

5017 **D.11.2 Fixed URI**

5018 /SceneMemberResURI

5019 **D.11.3 Resource Type**

5020 The resource type (rt) is defined as: oic.r.switch.binary.

5021 **D.11.4 RAML Definition**

```

5022 #%RAML 0.8
5023 title: Scene
5024 version: v1-20160622
5025 traits:
5026   - interface :
5027     queryParameters:
5028       if:
5029         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
5030
5031 /SceneMemberResURI:
5032   description: |
5033     Collection that models a sceneMember.
5034
5035   get:
5036     description: |
5037       Provides the scene member
5038
5039     responses :
5040       200:
5041         body:
5042           application/json:
5043             schema: /
5044               {
5045                 "$schema": "http://json-schema.org/draft-04/schema#",
5046                 "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5047 reserved.",
5048                 "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneMember-
5049 schema.json#",
5050                 "title" : "Scene Member",
5051                 "definitions": {
5052                   "oic.sceneMember": {

```



```

5053         "type": "object",
5054         "properties": {
5055             "n": {
5056                 "type": "string",
5057                 "description": "Used to name the Scene collection",
5058                 "format": "UTF8"
5059             },
5060             "id": {
5061                 "type": "string",
5062                 "description": "Can be an value that is unique to the use context or a
5063 UUIDv4"
5064             },
5065             "SceneMappings" : {
5066                 "type": "array",
5067                 "description": "array of mappings per scene, can be 1",
5068                 "items": [
5069                     {
5070                         "type": "object",
5071                         "properties": {
5072                             "scene": {
5073                                 "type": "string",
5074                                 "description": "Specifies a scene value that will acted upon"
5075                             },
5076                             "memberProperty": {
5077                                 "type": "string",
5078                                 "readOnly": true,
5079                                 "description": "property name that will be mapped"
5080                             },
5081                             "memberValue": {
5082                                 "type": "string",
5083                                 "readOnly": true,
5084                                 "description": "value of the Member Property"
5085                             }
5086                         },
5087                         "required": [ "scene", "memberProperty", "memberValue" ]
5088                     }
5089                 ]
5090             },
5091             "link": {
5092                 "type": "string",
5093                 "description": "web link that points at a resource",
5094                 "$ref": "oic.oic-link-schema.json#"
5095             },
5096             "required": [ "link" ]
5097         },
5098     },
5099     "type": "object",
5100     "allOf" : [
5101         { "$ref": "#/definitions/oic.sceneMember" }
5102     ]
5103 }
5104
5105 example: /
5106 {
5107     "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
5108     "n": "my binary switch (for light bulb) mappings",
5109     "link": { "href": "coap://mydevice/mybinaryswitch",
5110             "if": "oic.if.a",
5111             "rt": "oic.r.switch.binary" },
5112     "sceneMappings": [
5113         {
5114             "scene": "off",
5115             "memberProperty": "value",
5116             "memberValue": true
5117         },
5118         {
5119             "scene": "Reading",
5120

```

```

5123         "memberProperty": "value",
5124         "memberValue": false
5125     },
5126     {
5127         "scene": "TVWatching",
5128         "memberProperty": "value",
5129         "memberValue": true
5130     }
5131 ]
5132 }
5133

```

5134 D.11.5 Property Definition

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
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
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 A Resource

5135 D.11.6 CRUDN Behaviour

Resource	Create	Read	Update	Delete	Notify
/SceneMemberResURI		get			

5136 D.12 Resource directory resource

5137 D.12.1 Introduction

5138 Resource to be exposed by any Device that can act as a Resource Directory

5139 D.12.2 Fixed URI

5140 /oic/rd

5141 D.12.3 Resource Type

5142 The resource type (rt) is defined as: oic.wk.rd.

5143 D.12.4 RAML Definition

```

5144 #%RAML 0.8
5145 title: Resource Directory
5146 version: v1-20160622
5147 traits:
5148   - rddefinterface :

```

```

5149     queryParameters:
5150         if:
5151             description: Interface is optional since there is only one interface supported for the
5152 Resource Type
5153 Both for RD selectin and for publish
5154
5155             type: string
5156             enum: ["oic.if.baseline"]
5157             default: oic.if.baseline
5158
5159 /oic/rd:
5160     description: |
5161 Resource to be exposed by any Device that can act as a Resource Directory
5162
5163     get:
5164         description: |
5165 Get the attributes of the Resource Directory for selection purposes.
5166
5167         queryParameters:
5168             rt:
5169                 enum: oic.wk.rd
5170                 type: string
5171
5172             description: Only one Resource Type is used for GET; RT is optional
5173
5174             required: false
5175
5176             example: GET /oic/rd?rt=oic.wk.rd
5177
5178     responses :
5179         200:
5180             description: |
5181 Respond with the selector criteria - either the set of attributes or the bias factor
5182
5183             body:
5184                 application/json:
5185                     schema: /
5186
5187 {
5188     "$schema": "http://json-schema.org/draft-04/schema#",
5189     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5190 reserved.",
5191     "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.selection-
5192 schema.json#",
5193     "title" : "RD Selection",
5194     "definitions": {
5195         "oic.rd.attributes": {
5196             "type": "object",
5197             "properties": {
5198                 "n": {
5199                     "type": "string",
5200                     "description": "A human friendly name for the Resource Directory",
5201                     "format": "UTF8"
5202                 },
5203                 "di": {
5204                     "$ref": "oic.types-schema.json#/definitions/uuid",
5205                     "description": "A unique identifier for the Resource Directory - the same
5206 as the device ID of the RD"
5207                 },
5208                 "sel": {
5209                     "description": "Selection criteria that a device wanting to publish to any
5210 RD can use to choose this Resource Directory over others that are discovered",
5211                     "oneOf": [

```

```

5210         {
5211             "type": "object",
5212             "properties": {
5213                 "pwr": {
5214                     "type": "string",
5215                     "enum": [ "ac", "batt", "safe" ],
5216                     "description": "A hint about how the RD is powered. If AC then this
5217 is stronger than battery powered. If source is reliable (safe) then appropriate mechanism for
5218 managing power failure exists"
5219                 },
5220                 "conn": {
5221                     "type": "string",
5222                     "enum": [ "wrld", "wrlds" ],
5223                     "description": "A hint about the networking connectivity of the RD.
5224 *wrld* if wired connected and *wrlds* if wireless connected."
5225                 },
5226                 "bw": {
5227                     "type": "string",
5228                     "description": "Qualitative bandwidth of the connection",
5229                     "enum": [ "high", "low", "lossy" ]
5230                 },
5231                 "mf": {
5232                     "type": "integer",
5233                     "description": "Memory factor - Ratio of available memory to total
5234 memory expressed as a percentage"
5235                 },
5236                 "load": {
5237                     "type": "array",
5238                     "items": {
5239                         "type": "number"
5240                     },
5241                     "minitems": 3,
5242                     "maxitems": 3,
5243                     "description": "Current load capacity of the RD. Expressed as a
5244 load factor 3-tuple (upto two decimal points each). Load factor is based on request processed in a
5245 1 minute, 5 minute window and 15 minute window"
5246                 }
5247             },
5248             {
5249                 {
5250                     "type": "integer",
5251                     "minimum": 0,
5252                     "maximum": 100,
5253                     "description": "A bias factor calculated by the Resource directory -
5254 the value is in the range of 0 to 100 - 0 implies that RD is not to be selected. Client chooses RD
5255 with highest bias factor or randomly between RDs that have same bias factor"
5256                 }
5257             ]
5258         }
5259     }
5260 },
5261     "type": "object",
5262     "allOf": [ { "$ref": "#/definitions/oic.rd.attributes" } ],
5263     "required": [ "sel" ]
5264 }
5265
5266
5267     example: /
5268     {
5269         "rt": "oic.wk.rd",
5270         "sel": 50
5271     }
5272
5273     post:
5274         description: |
5275             Publish the resource information
5276             Appropriates parts of the information posted will be discovered through /oic/res
5277

```

```

5278     queryParameters:
5279         rt:
5280             enum: oic.wk.rdpub
5281             type: string
5282             description: Only one Resource Type is used for GET; RT is optional
5283
5284         required: false
5285         example: GET /oic/rd?rt=oic.wk.rdpub
5286
5287     body:
5288         application/json:
5289             schema: /
5290                 {
5291                     "$schema": "http://json-schema.org/draft-04/schema#",
5292                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5293 reserved.",
5294                     "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
5295 schema.json#",
5296                     "title": "RD Publish & Update",
5297                     "definitions": {
5298                         "oic.rd.publish": {
5299                             "description": "Publishes resources as OIC Links into the resource directory",
5300                             "properties": {
5301                                 "linkSet": {
5302                                     "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-tagged-
5303 setoflinks"
5304                                 },
5305                                 "ttl": {
5306                                     "type": "integer",
5307                                     "description": "Time to indicate a RD, how long to keep this published item.
5308 After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
5309 publishing device updates the ttl using the update schema"
5310                                 }
5311                             }
5312                         },
5313                     },
5314                     "type": "object",
5315                     "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
5316                     "required": [ "links" ],
5317                     "dependencies": {
5318                         "links": [ "ttl" ]
5319                     }
5320                 }
5321
5322     responses :
5323         200:
5324             description: |
5325                 Respond with the same schema as publish but with the links have the "ins" parameter set
5326 to the appropriate instance value.
5327                 This value is used by the receiver to manage that OIC Link instance.
5328
5329         body:
5330             application/json:
5331                 schema: /
5332                     {
5333                         "$schema": "http://json-schema.org/draft-04/schema#",
5334                         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5335 reserved.",
5336                         "id": "https://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
5337 schema.json#",
5338                         "title": "RD Publish & Update",
5339                         "definitions": {
5340                             "oic.rd.publish": {
5341                                 "description": "Publishes resources as OIC Links into the resource directory",

```

```

5342         "properties": {
5343             "linkSet": {
5344                 "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-
5345 tagged-setoflinks"
5346             },
5347             "ttl": {
5348                 "type": "integer",
5349                 "description": "Time to indicate a RD, how long to keep this published
5350 item. After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
5351 publishing device updates the ttl using the update schema"
5352             }
5353         }
5354     },
5355     "type": "object",
5356     "allOf": [{ "$ref": "#/definitions/oic.rd.publish" }],
5357     "required": [ "links" ],
5358     "dependencies": {
5359         "links": [ "ttl" ]
5360     }
5361 }
5362 }
5363
5364 example: /
5365 {
5366     "links": [
5367         {
5368             "href": "coap://someAuthority:1000/somePath",
5369             "rt": "oic.r.someResource",
5370             "if": "oic.if.a",
5371             "ins": 12345
5372         },
5373         {
5374             "href": "coap://someAuthority:1000/somePath",
5375             "rt": "oic.r.someOtherResource",
5376             "if": "oic.if.baseline",
5377             "ins": 54321
5378         }
5379     ],
5380     "ttl": 600
5381 }
5382
5383 delete:
5384 description: |
5385 Delete a particular OIC Link - the link may be a simple link or a link in a tagged set.
5386
5387 queryParameters:
5388 di:
5389 type: string
5390 description: This is used to determine which set of links to operate on. (Need
5391 authentication to ensure that there is no spoofing). If instance is omitted then the entire set of
5392 links from this device ID is deleted
5393
5394 required: true
5395 example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"
5396
5397 ins:
5398 type: string
5399 description: Instance of the link to delete
5400 Value of parameter is a string where instance to be deleted are comma separated
5401
5402 required: false
5403 example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1";ins="20"
5404

```

```

5405     responses :
5406         200:
5407             description: |
5408                 The delete succeeded
5409

```

5410 **D.12.5 Property Definition**

Property name	Value type	Mandatory	Access mode	Description
id			Read Write	
n	string		Read Write	A human friendly name for the Resource Directory
di			Read Write	A unique identifier for the Resource Directory - the same as the device ID of the RD
sel		yes	Read Write	
pwr	string		Read Write	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	string		Read Write	A hint about the networking connectivity of the RD. *wrđ* if wired connected and *wrls* if wireless connected.
bw	string		Read Write	Qualitative bandwidth of the connection
mf	integer		Read Write	Memory factor - Ratio of available memory to total memory expressed as a percentage
load	array		Read Write	

5411 **D.12.6 CRUDN Behaviour**

Resource	Create	Read	Update	Delete	Notify
/oic/rd		get	post	delete	

5412