

# OCF Core Specification

VERSION 2.0.5 | September 2019



**OPEN** CONNECTIVITY  
FOUNDATION™

CONTACT [admin@openconnectivity.org](mailto:admin@openconnectivity.org)

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

## Legal Disclaimer

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19

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

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

20 **CONTENTS**

21 1 Scope ..... 1

22 2 Normative references ..... 1

23 3 Terms, definitions, and abbreviated terms ..... 3

24 3.1 Terms and definitions..... 3

25 3.2 Abbreviated terms..... 6

26 4 Document conventions and organization..... 8

27 4.1 Conventions..... 8

28 4.2 Notation..... 8

29 4.3 Data types ..... 9

30 5 Architecture ..... 10

31 5.1 Overview ..... 10

32 5.2 Principle ..... 11

33 5.3 Functional block diagram ..... 12

34 5.4 Framework..... 14

35 6 Identification and addressing ..... 14

36 6.1 Introduction..... 14

37 6.2 Identification ..... 15

38 6.2.1 Overview ..... 15

39 6.2.2 Resource identification and addressing ..... 15

40 6.3 Namespace:..... 16

41 6.4 Network addressing ..... 16

42 7 Resource model ..... 17

43 7.1 Introduction..... 17

44 7.2 Resource ..... 18

45 7.3 Property..... 18

46 7.3.1 Introduction ..... 18

47 7.3.2 Common Properties ..... 19

48 7.4 Resource Type ..... 20

49 7.4.1 Introduction ..... 20

50 7.4.2 Resource Type Property ..... 21

51 7.4.3 Resource Type definition ..... 21

52 7.4.4 Multi-value "rt" Resource ..... 23

53 7.5 Device Type..... 23

54 7.6 OCF Interface ..... 24

55 7.6.1 Introduction ..... 24

56 7.6.2 OCF Interface Property..... 24

57 7.6.3 OCF Interface methods..... 25

58 7.7 Resource representation ..... 41

59 7.8 Structure..... 41

60 7.8.1 Introduction ..... 41

61 7.8.2 Resource Relationships ..... 41

62 7.8.3 Collections..... 46

63	7.8.4	Atomic Measurement .....	48
64	7.9	Query Parameters .....	50
65	7.9.1	Introduction .....	50
66	7.9.2	Use of multiple parameters within a query .....	50
67	7.9.3	Application to multi-value "rt" Resources .....	51
68	7.9.4	OCF Interface specific considerations for queries .....	51
69	8	CRUDN .....	52
70	8.1	Overview .....	52
71	8.2	CREATE .....	52
72	8.2.1	Overview .....	52
73	8.2.2	CREATE request .....	53
74	8.2.3	Processing by the Server .....	53
75	8.2.4	CREATE response .....	53
76	8.3	RETRIEVE .....	53
77	8.3.1	Overview .....	53
78	8.3.2	RETRIEVE request .....	54
79	8.3.3	Processing by the Server .....	54
80	8.3.4	RETRIEVE response .....	54
81	8.4	UPDATE .....	54
82	8.4.1	Overview .....	54
83	8.4.2	UPDATE request .....	54
84	8.4.3	Processing by the Server .....	55
85	8.4.4	UPDATE response .....	55
86	8.5	DELETE .....	55
87	8.5.1	Overview .....	55
88	8.5.2	DELETE request .....	56
89	8.5.3	Processing by the Server .....	56
90	8.5.4	DELETE response .....	56
91	8.6	NOTIFY .....	56
92	8.6.1	Overview .....	56
93	8.6.2	NOTIFICATION response .....	56
94	9	Network and connectivity .....	57
95	9.1	Introduction .....	57
96	9.2	Architecture .....	57
97	9.3	IPv6 network layer requirements .....	58
98	9.3.1	Introduction .....	58
99	9.3.2	IPv6 node requirements .....	59
100	10	OCF Endpoint .....	59
101	10.1	OCF Endpoint definition .....	59
102	10.2	OCF Endpoint information .....	60
103	10.2.1	Introduction .....	60
104	10.2.2	"ep" .....	60
105	10.2.3	"pri" .....	61
106	10.2.4	OCF Endpoint information in "eps" Parameter .....	61

107	10.3	OCF Endpoint discovery .....	62
108	10.3.1	Introduction .....	62
109	10.3.2	Implicit discovery .....	62
110	10.3.3	Explicit discovery with "/oic/res" response .....	62
111	11	Functional interactions .....	64
112	11.1	Introduction.....	64
113	11.2	Resource discovery .....	65
114	11.2.1	Introduction .....	65
115	11.2.2	Resource based discovery: mechanisms .....	65
116	11.2.3	Resource based discovery: Finding information .....	66
117	11.2.4	Resource discovery using "/oic/res" .....	72
118	11.2.5	Multicast discovery using "/oic/res" .....	73
119	11.3	Notification .....	74
120	11.3.1	Overview .....	74
121	11.3.2	Observe.....	74
122	11.4	Introspection.....	75
123	11.4.1	Overview .....	75
124	11.4.2	Usage of Introspection.....	78
125	12	Messaging.....	79
126	12.1	Introduction.....	79
127	12.2	Mapping of CRUDN to CoAP.....	80
128	12.2.1	Overview .....	80
129	12.2.2	URIs.....	80
130	12.2.3	CoAP method with request and response .....	80
131	12.2.4	Content-Format negotiation .....	82
132	12.2.5	OCF-Content-Format-Version information.....	82
133	12.2.6	Content-Format policy .....	83
134	12.2.7	CRUDN to CoAP response codes .....	84
135	12.2.8	CoAP block transfer.....	84
136	12.2.9	Generic requirements for CoAP multicast .....	84
137	12.3	Mapping of CRUDN to CoAP serialization over TCP .....	85
138	12.3.1	Overview .....	85
139	12.3.2	URIs.....	85
140	12.3.3	CoAP method with request and response .....	85
141	12.3.4	Content-Format negotiation .....	85
142	12.3.5	OCF-Content-Format-Version information.....	85
143	12.3.6	Content-Format policy .....	85
144	12.3.7	CRUDN to CoAP response codes .....	85
145	12.3.8	CoAP block transfer.....	85
146	12.3.9	Keep alive (connection health).....	85
147	12.4	Payload Encoding in CBOR .....	86
148	13	Security.....	86
149	Annex A (normative)	Resource Type definitions.....	87
150	A.1	List of Resource Type definitions .....	87

151	A.2	Atomic Measurement links list representation .....	87
152	A.2.1	Introduction .....	87
153	A.2.2	Example URI .....	87
154	A.2.3	Resource type .....	87
155	A.2.4	OpenAPI 2.0 definition.....	87
156	A.2.5	Property definition .....	94
157	A.2.6	CRUDN behaviour .....	95
158	A.3	Collection.....	95
159	A.3.1	Introduction .....	95
160	A.3.2	Example URI .....	95
161	A.3.3	Resource type .....	95
162	A.3.4	OpenAPI 2.0 definition.....	95
163	A.3.5	Property definition .....	103
164	A.3.6	CRUDN behaviour .....	104
165	A.4	Device .....	104
166	A.4.1	Introduction .....	104
167	A.4.2	Well-known URI .....	104
168	A.4.3	Resource type .....	104
169	A.4.4	OpenAPI 2.0 definition.....	104
170	A.4.5	Property definition .....	107
171	A.4.6	CRUDN behaviour .....	108
172	A.5	Introspection Resource .....	108
173	A.5.1	Introduction .....	108
174	A.5.2	Well-known URI .....	108
175	A.5.3	Resource type .....	108
176	A.5.4	OpenAPI 2.0 definition.....	108
177	A.5.5	Property definition .....	111
178	A.5.6	CRUDN behaviour .....	111
179	A.6	Platform .....	111
180	A.6.1	Introduction .....	111
181	A.6.2	Well-known URI .....	111
182	A.6.3	Resource type .....	111
183	A.6.4	OpenAPI 2.0 definition.....	112
184	A.6.5	Property definition .....	114
185	A.6.6	CRUDN behaviour .....	115
186	A.7	Discoverable Resources .....	115
187	A.7.1	Introduction .....	115
188	A.7.2	Well-known URI .....	115
189	A.7.3	Resource type .....	115
190	A.7.4	OpenAPI 2.0 definition.....	115
191	A.7.5	Property definition .....	119
192	A.7.6	CRUDN behaviour .....	120
193	Annex B (informative)	OpenAPI 2.0 Schema Extension.....	121
194	B.1	OpenAPI 2.0 Schema Reference.....	121

195	B.2	OpenAPI 2.0 Introspection empty file .....	121
196			
197			

198  
199  
200

## Figures

201	Figure 1 – Architecture - concepts .....	12
202	Figure 2 – Functional block diagram .....	13
203	Figure 3 – Communication layering model .....	14
204	Figure 4 – Example Resource .....	18
205	Figure 5 – CREATE operation.....	52
206	Figure 6 – RETRIEVE operation .....	53
207	Figure 7 – UPDATE operation.....	54
208	Figure 8 – DELETE operation .....	56
209	Figure 9 – High Level Network & Connectivity Architecture .....	58
210	Figure 10 – Resource based discovery: Finding information.....	66
211	Figure 11 – Observe Mechanism.....	74
212	Figure 12 – Example usage of oneOf JSON schema .....	77
213	Figure 13 – Interactions to check Introspection support and download the Introspection Device Data. ....	79
215	Figure 14 – Content-Format Policy for backward compatible OCF Clients negotiating lower OCF Content-Format-Version .....	84

217

## Tables

218  
219

220	Table 1 – Additional OCF Types .....	9
221	Table 2 – Name Property Definition .....	20
222	Table 3 – Resource Identity Property Definition .....	20
223	Table 4 – Resource Type Common Property definition.....	21
224	Table 5 – Example foobar Resource Type.....	22
225	Table 6 – Example foobar Properties .....	22
226	Table 7 – Resource Interface Property definition.....	24
227	Table 8 – OCF standard OCF Interfaces .....	25
228	Table 9 – Batch OCF Interface Example .....	30
229	Table 10 – "bm" Property definition.....	43
230	Table 11 – Resource Types Property definition .....	45
231	Table 12 – Mandatory Resource Types Property definition.....	46
232	Table 13 – Common Properties for Collections (in addition to Common Properties defined in 7.3.2) .....	47
234	Table 14 – Common Properties for Atomic Measurement (in addition to Common Properties defined in 7.3.2).....	48
236	Table 15 – Atomic Measurement Resource Type .....	50
237	Table 16 – Properties for Atomic Measurement (in addition to Common Properties defined in 7.3.2) .....	50

238



239	Table 17 – Parameters of CRUDN messages.....	52
240	Table 18 – "ep" value for Transport Protocol Suite.....	61
241	Table 19 – List of Core Resources.....	64
242	Table 20 – Mandatory discovery Core Resources.....	67
243	Table 21 – "oic.wk.res" Resource Type definition.....	67
244	Table 22 – Protocol scheme registry.....	68
245	Table 23 – "oic.wk.d" Resource Type definition.....	69
246	Table 24 – "oic.wk.p" Resource Type definition.....	71
247	Table 25 – Introspection Resource.....	78
248	Table 26 – "oic.wk.introspection" Resource Type definition.....	78
249	Table 27 – CoAP request and response.....	80
250	Table 28 – OCF Content-Formats.....	82
251	Table 29 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option	
252	Numbers.....	82
253	Table 30 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version	
254	Representation.....	83
255	Table 31 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-	
256	Version Representation.....	83
257	Table A-1 – Alphabetized list of Core Resources.....	87
258	Table A-2 – The Property definitions of the Resource with type "rt" =	
259	"oic.wk.atomicmeasurement"......	94
260	Table A-3 – The CRUDN operations of the Resource with type "rt" =	
261	"oic.wk.atomicmeasurement"......	95
262	Table A-4 – The Property definitions of the Resource with type "rt" = "oic.wk.col"......	103
263	Table A-5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col"......	104
264	Table A-6 – The Property definitions of the Resource with type "rt" = "oic.wk.d"......	107
265	Table A-7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d"......	108
266	Table A-8 – The Property definitions of the Resource with type "rt" =	
267	"oic.wk.introspection"......	111
268	Table A-9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".	111
269	Table A-10 – The Property definitions of the Resource with type "rt" = "oic.wk.p"......	114
270	Table A-11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p"......	115
271	Table A-12 – The Property definitions of the Resource with type "rt" = "None"......	120
272	Table A-13 – The CRUDN operations of the Resource with type "rt" = "None"......	120
273		
274		

## 275 **1 Scope**

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

- 277 – Core specification (this document): The Core specification document specifies the Framework,  
278 i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles  
279 implementation for Internet of Things (IoT) usages and ecosystems. This document is  
280 mandatory for all Devices to implement.
- 281 – Core optional specification: The Core optional specification document specifies the Framework,  
282 i.e., the OCF core architecture, interfaces, protocols and services to enable OCF profiles  
283 implementation for Internet of Things (IoT) usages and ecosystems that can optionally be  
284 implemented by any Device.
- 285 – Core extension specification(s): The Core extension specification(s) document(s) specifies  
286 optional OCF Core functionality that are significant in scope (e.g., Wi-Fi easy setup, Cloud).

## 287 **2 Normative references**

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

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

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

295 ISO/IEC 30118-2:2018, *Information technology – Open Connectivity Foundation (OCF)*  
296 *Specification – Part 2: Security specification*  
297 <https://www.iso.org/standard/74239.html>  
298 Latest version available at: [https://openconnectivity.org/specs/OCF\\_Security\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Security_Specification.pdf)

299 OCF Easy Wi-Fi Setup, *Information technology – Open Connectivity Foundation (OCF)*  
300 *Specification – Part 7: Wi-Fi Easy Setup specification*  
301 Latest version available at: [https://openconnectivity.org/specs/OCF\\_Wi-](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)  
302 [Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)

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

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

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

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

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

313 IETF RFC 4941, *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*, September  
314 2007  
315 <https://www.rfc-editor.org/info/rfc4941>

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

318 IETF RFC 5988, *Web Linking: General Syntax*, October 2010  
319 <https://www.rfc-editor.org/info/rfc5988>

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

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

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

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

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

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

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

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

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

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

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

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

347 IETF RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*, August  
348 2016  
349 <https://www.rfc-editor.org/info/rfc7959>

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

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

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

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

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

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

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

### 366 **3 Terms, definitions, and abbreviated terms**

#### 367 **3.1 Terms and definitions**

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

369 ISO and IEC maintain terminological databases for use in standardization at the following  
370 addresses:

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

##### 373 **3.1.1**

##### 374 **Atomic Measurement**

375 a design pattern that ensures that the Client (3.1.6) can only access the Properties (3.1.32) of  
376 linked Resources (3.1.30) atomically, that is as a single group

##### 377 **3.1.2**

##### 378 **Bridged Client**

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

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

##### 381 **3.1.3**

##### 382 **Bridged Device**

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

##### 384 **3.1.4**

##### 385 **Bridged Protocol**

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

##### 387 **3.1.5**

##### 388 **Bridged Server**

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

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

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

##### 392 **3.1.6**

##### 393 **Client**

394 a logical entity that accesses a Resource (3.1.30) on a Server (3.1.35)

##### 395 **3.1.7**

##### 396 **Collection**

397 a Resource (3.1.30) that contains zero or more Links (3.1.21)

398 **3.1.8**  
399 **Common Properties**  
400 Properties (3.1.32) specified for all Resources (3.1.30)

401 **3.1.9**  
402 **Composite Device**  
403 a Device (3.1.13) that is modelled as multiple Device Types (3.1.14); with each component Device  
404 Type (3.1.14) being exposed as a Collection (3.1.7)

405 **3.1.10**  
406 **Configuration Source**  
407 a cloud or service network or a local read-only file which contains and provides configuration  
408 related information to the Devices (3.1.13)

409 **3.1.11**  
410 **Core Resources**  
411 those Resources (3.1.30) that are defined in this document

412 **3.1.12**  
413 **Default OCF Interface**  
414 an OCF Interface (3.1.18) used to generate the response when an OCF Interface (3.1.18) is omitted  
415 in a request

416 **3.1.13**  
417 **Device**  
418 a logical entity that assumes one or more roles, e.g., Client (3.1.6), Server (3.1.35)

419 Note 1 to entry: More than one Device (3.1.13) can exist on a Platform (3.1.29).

420 **3.1.14**  
421 **Device Type**  
422 a uniquely named definition indicating a minimum set of Resource Types (3.1.33) that a Device  
423 (3.1.13) supports

424 Note 1 to entry: A Device Type (3.1.14) provides a hint about what the Device (3.1.13) is, such as a light or a fan, for  
425 use during Resource (3.1.30) discovery.

426 **3.1.15**  
427 **Discoverable Resource**  
428 a Resource (3.1.30) that is listed in "/oic/res"

429 **3.1.16**  
430 **OCF Endpoint**  
431 entity participating in the OCF protocol, further identified as the source or destination of a request  
432 and response messages for a given Transport Protocol Suite

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

434 **3.1.17**  
435 **Framework**  
436 a set of related functionalities and interactions defined in this document, which enable  
437 interoperability across a wide range of networked devices, including IoT

438 **3.1.18**  
439 **OCF Interface**  
440 interface description in accordance with IETF RFC 6690 and as defined by OCF that provides a  
441 view to and permissible responses from a Resource (3.1.30)

442 **3.1.19**  
443 **Introspection**  
444 mechanism to determine the capabilities of the hosted Resources (3.1.30) of a Device (3.1.13)

445 **3.1.20**  
446 **Introspection Device Data (IDD)**  
447 data that describes the payloads per implemented method of the Resources (3.1.30) that make up  
448 the Device (3.1.13)

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

450 **3.1.21**  
451 **Links**  
452 extends typed web links according to IETF RFC 5988

453 **3.1.22**  
454 **Non-Discoverable Resource**  
455 a Resource (3.1.30) that is not listed in "/oic/res"

456 Note 1 to entry: The Resource (3.1.30) can be reached by a Link (3.1.21) which is conveyed by another Resource  
457 (3.1.30). For example a Resource (3.1.30) linked in a Collection (3.1.7) does not have to be listed in "/oic/res", since  
458 traversing the Collection (3.1.7) would discover the Resource (3.1.30) implemented on the Device (3.1.13).

459 **3.1.23**  
460 **Notification**  
461 the mechanism to make a Client (3.1.6) aware of state changes in a Resource (3.1.30)

462 **3.1.24**  
463 **Observe**  
464 the act of monitoring a Resource (3.1.30) by sending a RETRIEVE operation which is cached by  
465 the Server (3.1.35) hosting the Resource (3.1.30) and reprocessed on every change to that  
466 Resource (3.1.30)

467 **3.1.25**  
468 **OpenAPI 2.0**  
469 Resource (3.1.30) and Introspection Device Data (3.1.20) definitions used in this document as  
470 defined in the OpenAPI specification

471 **3.1.26**  
472 **Parameter**  
473 an element that provides metadata about a Resource (3.1.30) referenced by the target URI of a  
474 Link (3.1.21)

475 **3.1.27**  
476 **Partial UPDATE**  
477 an UPDATE operation to a Resource (3.1.30) that includes a subset of the Properties (3.1.32) that  
478 are visible via the OCF Interface (3.1.18) being applied for the Resource Type (3.1.33)

479 **3.1.28**  
480 **Physical Device**  
481 the physical thing on which a Device(s) (3.1.13) is exposed

482 **3.1.29**  
483 **Platform**  
484 a Physical Device (3.1.28) containing one or more Devices (3.1.13)

485 **3.1.30**  
486 **Resource**  
487 represents an entity modelled and exposed by the Framework (3.1.17)

488 **3.1.31**  
489 **Resource Interface**  
490 a qualification of the permitted requests on a Resource (3.1.30)

491 **3.1.32**  
492 **Property**  
493 a significant aspect or Parameter (3.1.26) of a Resource (3.1.30), including metadata, that is  
494 exposed through the Resource (3.1.30)

495 **3.1.33**  
496 **Resource Type**  
497 a uniquely named definition of a class of Properties (3.1.32) and the interactions that are supported  
498 by that class

499 Note 1 to entry: Each Resource (3.1.30) has a Property (3.1.32) "rt" whose value is the unique name of the Resource  
500 Type (3.1.33).

501 **3.1.34**  
502 **Secure OCF Endpoint**  
503 an OCF Endpoint (3.1.16) with a secure connection (e.g., CoAPS)

504 **3.1.35**  
505 **Server**  
506 a Device (3.1.13) with the role of providing Resource (3.1.30) state information and facilitating  
507 remote interaction with its Resources (3.1.30)

508 **3.1.36**  
509 **Unsecure OCF Endpoint**  
510 an OCF Endpoint ( ) with an unsecure connection (e.g., CoAP)

511 **3.1.37**  
512 **Vertical Resource Type**  
513 a Resource Type (3.1.33) in a vertical domain specification

514 Note 1 to entry: An example of a Vertical Resource Type (3.1.37) would be "oic.r.switch.binary".

515 **3.1.38**  
516 **Virtual OCF Client**  
517 logical representation of a Bridged Client (3.1.2), which an Bridged Device (3.1.3) exposes to  
518 Servers (3.1.35)

519 **3.1.39**  
520 **Virtual OCF Device (or VOD)**  
521 Virtual OCF Client (3.1.38) or Virtual OCF Server (3.1.40)

522 **3.1.40**  
523 **Virtual OCF Server**  
524 logical representation of a Bridged Server (3.1.5), which an Bridged Device (3.1.3) exposes to  
525 Clients (3.1.6)

526 **3.2 Abbreviated terms**

527 **3.2.1**  
528 **ACL**  
529 Access Control List

530 Note 1 to entry: The details are defined in ISO/IEC 30118-2:2018.

531 **3.2.2**  
532 **BLE**  
533 Bluetooth Low Energy

534 **3.2.3**  
535 **CBOR**  
536 Concise Binary Object Representation

537 **3.2.4**  
538 **CoAP**  
539 Constrained Application Protocol

540 **3.2.5**  
541 **CoAPS**  
542 Secure Constrained Application Protocol

543 **3.2.6**  
544 **DTLS**  
545 Datagram Transport Layer Security

546 Note 1 to entry: The details are defined in IETF RFC 6347.

547 **3.2.7**  
548 **EXI**  
549 Efficient XML Interchange

550 **3.2.8**  
551 **IP**  
552 Internet Protocol

553 **3.2.9**  
554 **IRI**  
555 Internationalized Resource Identifiers

556 **3.2.10**  
557 **ISP**  
558 Internet Service Provider

559 **3.2.11**  
560 **JSON**  
561 JavaScript Object Notation

562 **3.2.12**  
563 **mDNS**  
564 Multicast Domain Name Service

565 **3.2.13**  
566 **MTU**  
567 Maximum Transmission Unit

568 **3.2.14**  
569 **NAT**  
570 Network Address Translation

571 **3.2.15**  
572 **OCF**  
573 Open Connectivity Foundation



574 the organization that created this document

575 **3.2.16**

576 **REST**

577 Representational State Transfer

578 **3.2.17**

579 **RESTful**

580 REST-compliant Web services

581 **3.2.18**

582 **UDP**

583 User Datagram Protocol

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

585 **3.2.19**

586 **URI**

587 Uniform Resource Identifier

588 **3.2.20**

589 **URN**

590 Uniform Resource Name

591 **3.2.21**

592 **UTC**

593 Coordinated Universal Time

594 **3.2.22**

595 **UUID**

596 Universal Unique Identifier

597 **3.2.23**

598 **XML**

599 Extensible Markup Language

600 **4 Document conventions and organization**

601 **4.1 Conventions**

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

606 **4.2 Notation**

607 In this document, features are described as required, recommended, allowed or DEPRECATED as  
608 follows:

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

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

613 Recommended (or should)(S).

614 – These features add functionality supported by Core Architecture and should be implemented.  
615 Recommended features take advantage of the capabilities Core Architecture, usually without

616 imposing major increase of complexity. Notice that for compliance testing, if a recommended  
617 feature is implemented, it shall meet the specified requirements to be in compliance with these  
618 guidelines. Some recommended features could become requirements in the future. The phrase  
619 "should not" indicates behaviour that is permitted but not recommended.

620 Allowed (may or allowed)(O).

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

624 DEPRECATED.

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

631 Conditionally allowed (CA).

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

634 Conditionally required (CR).

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

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

639 Words that are emphasized are printed in italic.

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

#### 644 **4.3 Data types**

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

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

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

655 **Table 1 – Additional OCF Types**

Pattern Name	Pattern	Description
"csv"	<none>	A comma separated list of values encoded within a string. The value type in the csv is described by the

		Property where the csv is used. For example a csv of integers.  NOTE csv is considered deprecated and an array of strings should be used instead for new Resources.
"date"	^([0-9]{4})-(1[0-2] 0[1-9])-(3[0-1] 2[0-9] 1[0-9] 0[1-9])\$	The full-date format pattern according to IETF RFC 3339
"duration"	^(P(?:!\$)([0-9]+Y)?([0-9]+M)?([0-9]+W)?([0-9]+D)?((T(?:=[0-9]+[HMS])([0-9]+H)?([0-9]+M)?([0-9]+S)?)?))\$ ^P([0-9]+W)\$ ^P([0-9]{4})-(1[0-2] 0[1-9])-(3[0-1] 2[0-9] 1[0-9] 0[1-9])T(2[0-3] 1[0-9] 0[1-9]):([0-5][0-9]):([0-5][0-9])\$ ^P([0-9]{4})(1[0-2] 0[1-9])(3[0-1] 2[0-9] 1[0-9] 0[1-9])T(2[0-3] 1[0-9] 0[1-9])([0-5][0-9])([0-5][0-9])\$	A string representing duration formatted as defined in ISO 8601. Allowable formats are: P[n]Y[n]M[n]DT[n]H[n]M[n]S, P[n]W, P[n]Y[n]-M[n]-DT[0-23]H[0-59]:M[0-59]:S, and P[n]W, P[n]Y[n]M[n]DT[0-23]H[0-59]M[0-59]S. P is mandatory, all other elements are optional, time elements must follow a T.
"int64"	^0(?:-[1-9][0-9]{0,18})\$	A string instance is valid against this attribute if it contains an integer in the range $[-(2^{63}), (2^{63})-1]$  NOTE IETF RFC 7159 clause 6 explains that JSON integers outside the range $[-(2^{53})+1, (2^{53})-1]$ are not interoperable and so JSON numbers cannot be used for 64-bit numbers.
"language-tag"	^[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*\$	An IETF language tag formatted according to IETF RFC 5646 clause 2.1.
"uint64"	^0?([1-9][0-9]{0,19})\$	A string instance is valid against this attribute if it contains an integer in the range $[0, (2^{64})-1]$  Also see note for "int64"
"uuid"	^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}\$	A UUID string representation formatted according to IETF RFC 4122 clause 3.

656

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

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

## 661 5 Architecture

### 662 5.1 Overview

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

667 Specifically, the architecture provides:

- 668 – A communication and interoperability framework for multiple market segments (Consumer,  
669 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,  
670 transports and use cases.
- 671 – A common and consistent model for describing the environment and enabling information and  
672 semantic interoperability.
- 673 – Common communication protocols for discovery and connectivity.
- 674 – Common security and identification mechanisms.
- 675 – Opportunity for innovation and product differentiation.
- 676 – A scalable solution addressing different Device capabilities, applicable to smart devices as well  
677 as the smallest connected things and wearable devices.

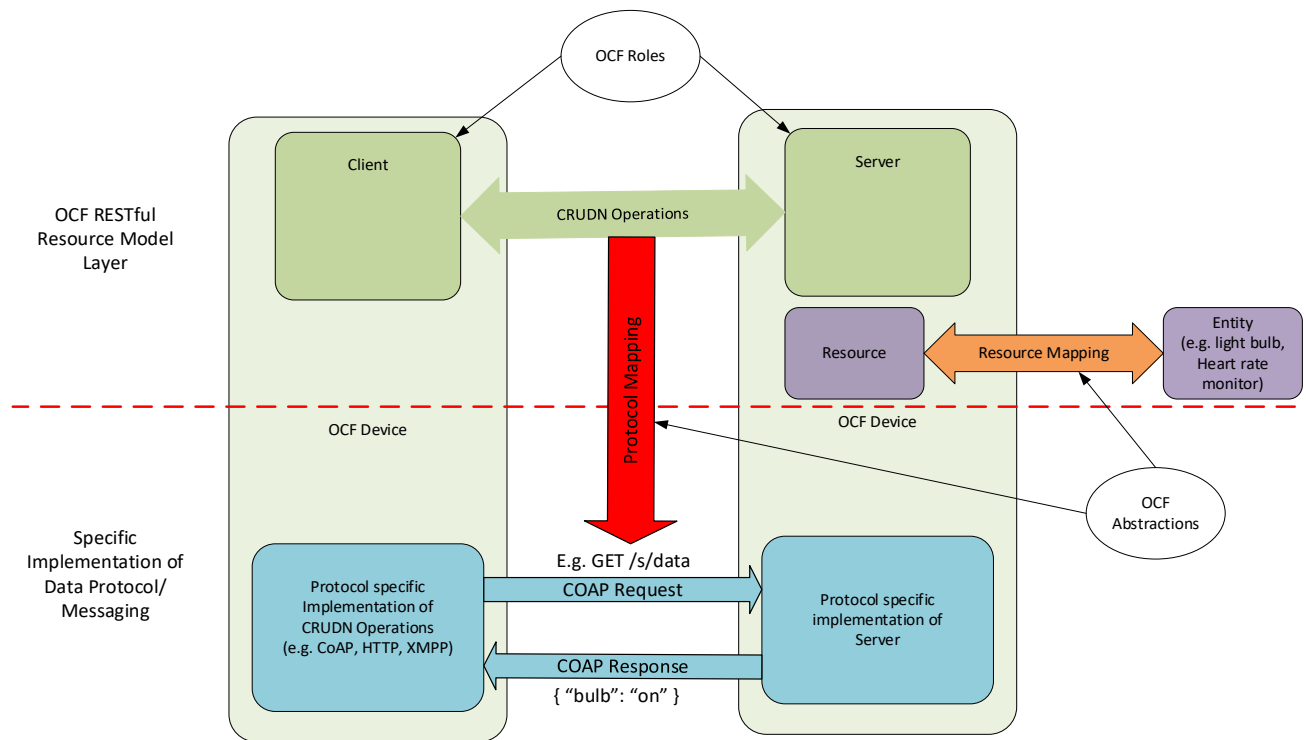
678 The architecture is based on the Resource Oriented Architecture design principles and described  
679 in the 5.2 through 5.4 respectively. 5.2 presents the guiding principles for OCF operations. 5.3  
680 defines the functional block diagram and Framework.

## 681 **5.2 Principle**

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

686 The architecture defines the overall structure of the Framework as an information system and the  
687 interrelationships of the Entities that make up OCF. Entities are exposed as Resources, with their  
688 unique identifiers (URIs) and support interfaces that enable RESTful operations on the Resources.  
689 Every RESTful operation has an initiator of the operation (the Client) and a responder to the  
690 operation (the Server). In the Framework, the notion of the Client and Server is realized through  
691 roles. Any Device can act as a Client and initiate a RESTful operation on any Device acting as a  
692 Server. Likewise, any Device that exposes Entities as Resources acts as a Server. Conformant to  
693 the REST architectural style, each RESTful operation contains all the information necessary to  
694 understand the context of the interaction and is driven using a small set of generic operations, i.e.,  
695 CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY (CRUDN) defined in clause 8, which include  
696 representations of Resources.

697 Figure 1 depicts the architecture.



698  
699 **Figure 1 – Architecture - concepts**

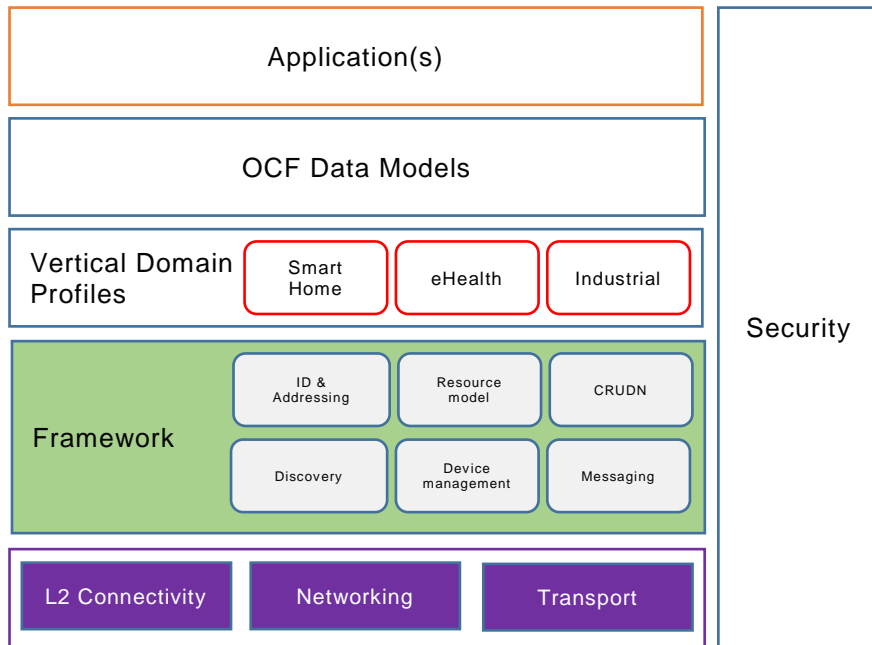
700 The architecture is organized conceptually into three major aspects that provide overall separation  
701 of concern: Resource model, RESTful operations and abstractions.

- 702 – Resource model: The Resource model provides the abstractions and concepts required to  
703 logically model, and logically operate on the application and its environment. The Core  
704 Resource model is common and agnostic to any specific application domain such as smart  
705 home, industrial or automotive. For example, the Resource model defines a Resource which  
706 abstracts an entity and the representation of a Resource maps the entity's state. Other  
707 Resource model concepts can be used to model other aspects, for example behaviour.
- 708 – RESTful operations: The generic CRUEN operations are defined using the RESTful paradigm  
709 to model the interactions with a Resource in a protocol and technology agnostic way. The  
710 specific communication or messaging protocols are part of the protocol abstraction and  
711 mapping of Resources to specific protocols is provided in 11.4.
- 712 – Abstraction: The abstractions in the Resource model and the RESTful operations are mapped  
713 to concrete elements using abstraction primitives. An entity handler is used to map an entity to  
714 a Resource and connectivity abstraction primitives are used to map logical RESTful operations  
715 to data connectivity protocols or technologies. Entity handlers may also be used to map  
716 Resources to Entities that are reached over protocols that are not natively supported by OCF.

717 **5.3 Functional block diagram**

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

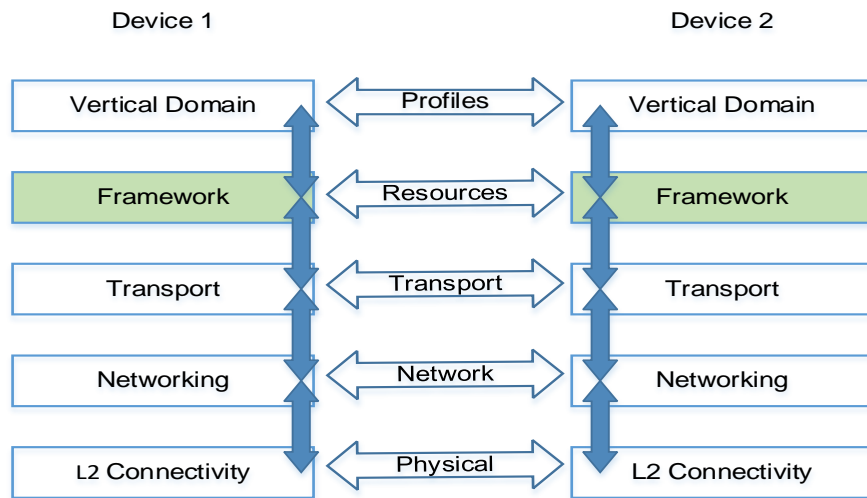
721



722

**Figure 2 – Functional block diagram**

- 723 – *L2 connectivity*: Provides the functionalities required for establishing physical and data link  
724 layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.
- 725 – *Networking*: Provides functionalities required for Devices to exchange data among themselves  
726 over the network (e.g., Internet).
- 727 – *Transport*: Provides end-to-end flow transport with specific QoS constraints. Examples of a  
728 transport protocol include TCP and UDP or new Transport protocols under development in the  
729 IETF, e.g., Delay Tolerant Networking (DTN).
- 730 – *Framework*: Provides the core functionalities as defined in this document. The functional block  
731 is the source of requests and responses that are the content of the communication between  
732 two Devices.
- 733 – *Vertical Domain profile*: Provides market segment specific functionalities, e.g., functions for the  
734 smart home market segment.
- 735 When two Devices communicate with each other, each functional block in a Device interacts with  
736 its counterpart in the peer Device as shown in Figure 3.



737  
738 **Figure 3 – Communication layering model**

739 **5.4 Framework**

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

- 741 – *Identification and addressing*. Defines the identifier and addressing capability. The Identification and addressing function is defined in clause 6.
- 742
- 743 – *Discovery*. Defines the process for discovering available.
- 744 – Devices (OCF Endpoint Discovery in clause 10) and
- 745 – Resources (Resource discovery in 11.2).
- 746 – *Resource model*. Specifies the capability for representation of entities in terms of Resources and defines mechanisms for manipulating the Resources. The Resource model function is defined in clause 7.
- 747
- 748
- 749 – *CRUDN*. Provides a generic scheme for the interactions between a Client and Server as defined in clause 8.
- 750
- 751 – *Messaging*. Provides specific message protocols for RESTful operation, i.e. CRUDN. For example, CoAP is a primary messaging protocol. The messaging function is defined in 12.
- 752
- 753 – *Security*. Includes authentication, authorization, and access control mechanisms required for secure access to Entities. The security function is defined in clause 13.
- 754

755 **6 Identification and addressing**

756 **6.1 Introduction**

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

758

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

760

761

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

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

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

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

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

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

## 780 **6.2 Identification**

### 781 **6.2.1 Overview**

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

### 796 **6.2.2 Resource identification and addressing**

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

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

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

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

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

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

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



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

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

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

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

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

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

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

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

### 839 **6.3 Namespace:**

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

### 843 **6.4 Network addressing**

844 The following are the addresses used in this document:

845 IP address

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

## 849 7 Resource model

### 850 7.1 Introduction

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

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

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

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

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

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

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

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

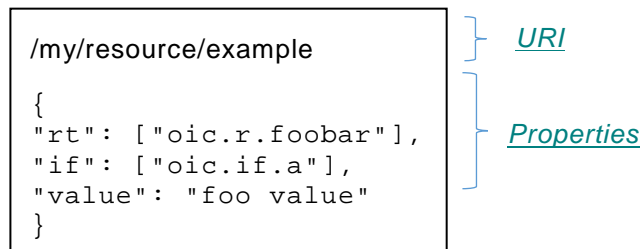
884 The information for Request or Response with the Representation may be communicated on the  
885 wire by serializing using a transfer protocol or encapsulated in the payload of the transport protocol  
886 – the specific method is determined by the normative mapping of the Request or Response to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

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

891 **7.2 Resource**

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

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



898

899 **Figure 4 – Example Resource**

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

904 **7.3 Property**

905 **7.3.1 Introduction**

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

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

914 In addition, the Property definition shall have a

- 915 – *Value Type* – the Value Type defines the values that a Property Value may take. The Value  
916 Type may be a simple data type (e.g. string, Boolean) as defined in 4.3 or may be a complex  
917 data type defined with a schema. The Value Type may define
  - 918 – Value Rules define the rules for the set of values that the Property Value may take. Such  
919 rules may define the range of values, the min-max, formulas, the set of enumerated values,  
920 patterns, conditional values, and even dependencies on values of other Properties. The  
921 rules may be used to validate the specific values in a Property Value and flag errors.
- 922 – *Mandatory* – specifies if the Property is mandatory or not for a given Resource Type.
- 923 – *Access modes* – specifies whether the Property may be read, written or both. Updates are  
924 equivalent to a write. "r" is used for read and "w" is used for write – both may be specified.  
925 Write does not automatically imply read.

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

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

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

## 934 **7.3.2 Common Properties**

### 935 **7.3.2.1 Introduction**

936 The Common Properties defined in this clause may be specified for all Resources. The following  
937 Properties are defined as Common Properties: Resource Type, Resource Interface, Name, and  
938 Resource Identity.

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

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

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

- 957 – *Resource Type* ("rt") – this Property is used to declare the Resource Type of that Resource.  
958 Since a Resource could be define by more than one Resource Type the Property Value of the  
959 Resource Type Property can be used to declare more than one Resource type (see clause  
960 7.4.4). See 7.3.2.3 for details.
- 961 – *OCF Interface* ("if") – this Property declares the OCF Interfaces supported by the Resource.  
962 The Property Value of the OCF Interface Property can be multi-valued and lists all the OCF  
963 Interfaces supported. See 7.3.2.4 for details.
- 964 – *Name* ("n") – the Property declares human-readable name assigned to the Resource. See  
965 7.3.2.5.
- 966 – *Resource Identity* ("id"): its Property Value shall be a unique (across the scope of the host  
967 Server) instance identifier for a specific instance of the Resource. The encoding of this identifier  
968 is Device and implementation dependent. See 7.3.2.6 for details.

### 969 **7.3.2.2 Property Name and Property Value definitions**

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

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

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

### 976 7.3.2.3 Resource Type

977 Resource Type Property is specified in 7.4.

### 978 7.3.2.4 OCF Interface

979 OCF Interface Property is specified in 7.6.

### 980 7.3.2.5 Name

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

983 **Table 2 – Name Property Definition**

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

984

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

### 987 7.3.2.6 Resource Identity

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

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

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

994

## 995 7.4 Resource Type

### 996 7.4.1 Introduction

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

999 The Resource Types of a Resource is declared using the Resource Type Common Property as  
1000 described in 7.3.2.3 or in a Link using the Resource Type Parameter.

1001 A Resource Type may either be pre-defined by OCF or in custom definitions by manufacturers, end  
1002 users, or developers of Devices (vendor-defined Resource Types). Resource Types and their  
1003 definition details may be communicated out of band (i.e. in documentation) or be defined explicitly  
1004 using a meta-language which may be downloaded and used by APIs or applications. OCF has

1005 adopted OpenAPI 2.0 as the specification method for OCF's RESTful interfaces and Resource  
1006 definitions.

1007 Every Resource Type shall be identified with a Resource Type ID which shall be represented using  
1008 the requirements and ABNF governing the Resource Type attribute in IETF RFC 6690 (clause 2 for  
1009 ABNF and clause 3.1 for requirements) with the caveat that segments are separated by a "."  
1010 (period). The entire string represents the Resource Type ID. When defining the ID each segment  
1011 may represent any semantics that are appropriate to the Resource Type. For example, each  
1012 segment could represent a namespace. Once the ID has been defined, the ID should be used  
1013 opaquely and implementations should not infer any information from the individual segments. The  
1014 string "oic", when used as the first segment in the definition of the Resource Type ID, is reserved  
1015 for OCF-defined Resource Types. All OCF defined Resource Types are to be registered with the  
1016 IANA Core Parameters registry as described also in IETF RFC 6690.

### 1017 7.4.2 Resource Type Property

1018 A Resource when instantiated or created shall have one or more Resource Types that are the  
1019 template for that Resource. The Resource Types that the Resource conforms to shall be declared  
1020 using the "rt" Common Property for the Resource as defined in Table 4. The Property Value for the  
1021 "rt" Common Property shall be the list of Resource Type IDs for the Resource Types used as  
1022 templates (i.e., "rt"=<list of Resource Type IDs>).

1023 **Table 4 – Resource Type Common Property definition**

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

1024  
1025 Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and  
1026 the host (i.e. Server) of the Resource.

### 1027 7.4.3 Resource Type definition

1028 Resource Type is specified as follows:

- 1029 – *Pre-defined URI* (optional) – a pre-defined URI may be specified for a specific Resource Type  
1030 in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that  
1031 Resource Type shall use only the pre-defined URI. An instance of a different Resource Type  
1032 shall not use the pre-defined URI.
- 1033 – *Resource Type Title* (optional) – a human friendly name to designate the Resource Type.
- 1034 – *Resource Type ID* – the value of "rt" Property which identifies the Resource Type, (e.g.,  
1035 "oic.wk.p").
- 1036 – *Resource Interfaces* – list of the OCF Interfaces that may be supported by the Resource Type.
- 1037 – *Properties* – definition of all the Properties that apply to the Resource Type. The Resource Type  
1038 definition shall define whether a property is mandatory, conditional mandatory, or optional.
- 1039 – *Related Resource Types* (optional) – the specification of other Resource Types that may be  
1040 referenced as part of the Resource Type, applicable to Collections.
- 1041 – *Mime Types* (optional) – mime types supported by the Resource including serializations (e.g.,  
1042 application/cbor, application/json, application/xml).

1043 Table 5 and Table 6 provides an example description of an illustrative foobar Resource Type and  
 1044 its associated Properties.

1045 **Table 5 – Example foobar Resource Type**

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

1046

1047 **Table 6 – Example foobar Properties**

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

1048

1049 For example, an instance of the foobar Resource Type.

```
1050 {
1051   "rt": ["oic.r.foobar"],
1052   "if": ["oic.if.a"],
1053   "value": "foo value"
1054 }
```

1055

1056 For example, a schema representation for the foobar Resource Type.

```
1057 {
1058   "$schema": "http://json-schema.org/draft-04/schema",
1059   "type": "object",
1060   "properties": {
1061     "rt": {
1062       "type": "array",
1063       "items": {
1064         "type": "string",
1065         "maxLength": 64
1066       },
1067       "minItems": 1,
1068       "readOnly": true,
1069       "description": "Resource Type of the Resource"
1070     },
1071     "if": {
1072       "type": "array",
1073       "items": {
1074         "type": "string",
1075         "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw",
1076 "oic.if.r", "oic.if.a", "oic.if.s"]
1077       },
1078       "value": {"type": "string"}
1079     },

```

```
1080     "required": ["rt", "if", "value"]
1081 }
```

#### 1082 **7.4.4 Multi-value "rt" Resource**

1083 Multi-value "rt" Resource means a Resource with multiple Resource Types where none of the  
1084 included Resource Types denote a well-known Resource Type (i.e. "oic.wk.<thing>"). Such a  
1085 Resource is associated with multiple Resource Types and so has an "rt" Property Value of multiple  
1086 Resource Type IDs (e.g. "rt": ["oic.r.switch.binary", "oic.r.light.brightness"]). The order of the  
1087 Resource Type IDs in the "rt" Property Value is meaningless. For example, "rt":  
1088 ["oic.r.switch.binary", "oic.r.light.brightness"] and "rt": ["oic.r.light.brightness", "oic.r.switch.binary"]  
1089 have the same meaning.

1090 Resource Types for multi-value "rt" Resources shall satisfy the following conditions:

- 1091 – Property Name – Property Names for each Resource Type shall be unique (within the scope of  
1092 the multi-value "rt" Resource) with the exception of Common Properties, otherwise there will be  
1093 conflicting Property semantics. If two Resource Types have a Property with the same Property  
1094 "Name, a multi-value "rt" Resource shall not be composed of these Resource Types.

1095 A multi-value "rt" Resource satisfies all the requirements for each Resource Type and conforms to  
1096 the OpenAPI 2.0 definitions for each component Resource Type. Thus the mandatory Properties  
1097 of a multi-value "rt" Resource shall be the union of all the mandatory Properties of each Resource  
1098 Type. For example, mandatory Properties of a Resource with "rt": ["oic.r.switch.binary",  
1099 "oic.r.light.brightness"] are "value" and "brightness", where the former is mandatory for  
1100 "oic.r.switch.binary" and the latter for "oic.r.light.brightness".

1101 The multi-value "rt" Resource Interface set shall be the union of the sets of OCF Interfaces from  
1102 the component Resource Types. The Resource Representation in response to a CRUDN action on  
1103 an OCF Interface shall be the union of the schemas that are defined for that OCF Interface. The  
1104 Default OCF Interface for a multi-value "rt" Resource shall be the baseline OCF Interface  
1105 ("oic.if.baseline") as that is the only guaranteed common OCF Interface between the Resource  
1106 Types.

1107 For clarity if each Resource Type supports the same set of OCF Interfaces, then the resultant multi-  
1108 value "rt" Resource has that same set of OCF Interfaces with a Default OCF Interface of baseline  
1109 ("oic.if.baseline").

1110 See 7.9.3 for the handling of query parameters as applied to a multi-value "rt" Resource.

#### 1111 **7.5 Device Type**

1112 A Device Type is a class of Device. Each Device Type defined will include a list of minimum  
1113 Resource Types that a Device shall implement for that Device Type. A Device may expose  
1114 additional standard and vendor defined Resource Types beyond the minimum list. The Device Type  
1115 is used in Resource discovery as specified in 11.2.3.

1116 Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in a  
1117 Link using the Resource Type Parameter.

1118 A Device Type may either be pre-defined by an ecosystem that builds on this document, or in  
1119 custom definitions by manufacturers, end users, or developers of Devices (vendor-defined Device  
1120 Types). Device Types and their definition details may be communicated out of band (like in  
1121 documentation).

1122 Every Device Type shall be identified with a Resource Type ID using the same syntax constraints  
1123 as a Resource Type.



1124 **7.6 OCF Interface**

1125 **7.6.1 Introduction**

1126 An OCF Interface provides first a view into the Resource and then defines the requests and  
1127 responses permissible on that view of the Resource. So this view provided by an OCF Interface  
1128 defines the context for requests and responses on a Resource. Therefore, the same request to a  
1129 Resource when targeted to different OCF Interfaces may result in different responses.

1130 An OCF Interface may be defined by either this document (a Core OCF Interface), manufacturers,  
1131 end users or developers of Devices (a vendor-defined OCF Interface).

1132 The OCF Interface Property lists all the OCF Interfaces the Resource support. All Resources shall  
1133 have at least one OCF Interface. The Default OCF Interface shall be defined by the Resource Type  
1134 definition. The Default OCF Interface associated with all OCF-defined Resource Types shall be the  
1135 supported OCF Interface listed first within the *applicable enumeration* in the definition of the  
1136 Resource Type (see Annex A for the OCF-defined Resource Types defined in this document). The  
1137 *applicable enumeration* is in the "parameters" enumeration referenced from the first "get" method  
1138 in the first "path" in the OpenAPI 2.0 file ("post" method if no "get" exists) for the Resource Type.  
1139 All Default OCF Interfaces specified in an OCF specification shall be mandatory.

1140 In addition to any defined OCF Interface in this document, all Resources shall support the baseline  
1141 OCF Interface ("oic.if.baseline") as defined in 7.6.3.2.

1142 See 7.9.4 for the use of queries to enable selection of a specific OCF Interface in a request.

1143 An OCF Interface may accept more than one media type. An OCF Interface may respond with more  
1144 than one media type. The accepted media types may be different from the response media types.  
1145 The media types are specified with the appropriate header parameters in the transfer protocol.  
1146 (NOTE: This feature has to be used judiciously and is allowed to optimize representations on the  
1147 wire) Each OCF Interface shall have at least one media type.

1148

1149 **7.6.2 OCF Interface Property**

1150 The OCF Interfaces supported by a Resource shall be declared using the OCF Interface Common  
1151 Property (Table 7), e.g., "if": ["oic.if.ll", "oic.if.baseline"]. The Property Value of an OCF Interface  
1152 Property shall be a lower case string with segments separated by a "." (dot). The string "oic", when  
1153 used as the first segment in the OCF Interface Property Value, is reserved for OCF-defined OCF  
1154 Interfaces. The OCF Interface Property Value may also be a reference to an authority similar to  
1155 IANA that may be used to find the definition of an OCF Interface. A Resource Type shall support  
1156 one or more of the OCF Interfaces defined in 7.6.3.

1157 **Table 7 – Resource Interface Property definition**

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

1158

1159 **7.6.3 OCF Interface methods**

1160 **7.6.3.1 Overview**

1161 OCF Interface methods shall not violate the defined OpenAPI 2.0 definitions for the Resources as  
1162 defined in Annex A.

1163 The defined OCF Interfaces are listed in Table 8:

1164 **Table 8 – OCF standard OCF Interfaces**

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

1165

1166 **7.6.3.2 Baseline OCF Interface**

1167 **7.6.3.2.1 Overview**

1168 The Representation that is visible using the baseline OCF Interface includes all the Properties of  
1169 the Resource including the Common Properties. The baseline OCF Interface shall be defined for  
1170 all Resource Types. All Resources shall support the baseline OCF Interface.

---

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

### 1171 **7.6.3.2.2 Use of RETRIEVE**

1172 The baseline OCF Interface is used when a Client wants to retrieve all Properties of a Resource;  
1173 that is the Server shall respond with a Resource representation that includes all of the implemented  
1174 Properties of the Resource. When the Server is unable to send back the whole Resource  
1175 representation, it shall reply with an error message. The Server shall not return a partial Resource  
1176 representation.

1177 An example response to a RETRIEVE request using the baseline OCF Interface:

```
1178 {  
1179   "rt": ["oic.r.temperature"],  
1180   "if": ["oic.if.a", "oic.if.baseline"],  
1181   "temperature": 20,  
1182   "units": "C",  
1183   "range": [0,100]  
1184 }
```

### 1185 **7.6.3.2.3 Use of UPDATE**

1186 Support for the UPDATE operation using the baseline OCF Interface should not be provided by a  
1187 Resource Type. Where a Resource Type needs to support the ability to be UPDATED this should  
1188 only be supported using one of the other OCF Interfaces defined in Table 8 that supports the  
1189 UPDATE operation.

1190 If a Resource Type is required to support UPDATE using the baseline OCF Interface, then all  
1191 Properties of a Resource with the exception of Common Properties may be modified using an  
1192 UPDATE operation only if the Resource Type defines support for UPDATE using baseline in the  
1193 applicable OpenAPI 2.0 schema for the Resource Type. If the OCF Interfaces exposed by a  
1194 Resource in addition to the baseline OCF Interface do not support the UPDATE operation, then  
1195 UPDATE using the baseline OCF Interface shall not be supported.

## 1196 **7.6.3.3 Links List OCF Interface**

### 1197 **7.6.3.3.1 Overview**

1198 The links list OCF Interface provides a view into the list of Links in a Resource. The Representation  
1199 visible through this OCF Interface has only the Links exposed as Property(-ies) that is(are) an array  
1200 (or arrays) of Links by the Resource – so this OCF Interface is used to manipulate or interact with  
1201 the list of Links. The Links list may be RETRIEVED using this OCF Interface.

1202 The links list OCF Interface is defined as follows:

- 1203 – The links list OCF Interface name is "oic.if.ll".
- 1204 – If there are no Links present in a Resource, then an empty list shall be returned in response to  
1205 a RETRIEVE request using the links list OCF Interface.
- 1206 – The Representation determined by this OCF Interface depends on the requesting Client. For a  
1207 Client that includes an OCF-Accept-Content-Format-Version option as defined in 12.2.5 in the  
1208 request the response only includes the Property value(s) of the Property(-ies) that are arrays  
1209 of Links, hence a Collection or "/oic/res" response with oic.if.ll is an array of Links.
- 1210 – The array of Links may be observed by a Client using the links list OCF Interface (i.e. by  
1211 following the procedures in clause 11.3.2 with the addition of a query parameter of "?if=oic.if.ll").
- 1212 – Any CREATE, UPDATE, or DELETE operation on any Link in the array of Links shall result in  
1213 the complete Resource representation for the links list OCF Interface as defined for the target  
1214 Resource (i.e. the full array of Links) subject to any applied filtering being provided in the  
1215 notification that is sent to the Client that initiated the Observe request.
- 1216 – If the act of deleting a Link results in no Links being present then an empty list shall be sent in  
1217 a notification.

### 1218 7.6.3.3.2 Example: links list OCF Interface

1219 A request to a Collection, where the request is to RETRIEVE the Links in room (the Links could be  
1220 referencing lights, fans, electric sockets etc).

1221 GET ocf://<devID>/a/room/1?if=oic.if.ll

1222 The response would be the array of OCF Links

```
1223  
1224 [  
1225   {  
1226     "href": "/the/light/1",  
1227     "rt": ["oic.r.switch.binary"],  
1228     "if": ["oic.if.a", "oic.if.baseline"],  
1229     "eps": [  
1230       {"ep": "coaps://[2001:db8:a::b1d4]:55555"}]  
1231     },  
1232     {  
1233       "href": "/the/light/2",  
1234       "rt": ["oic.r.switch.binary"],  
1235       "if": ["oic.if.a", "oic.if.baseline"],  
1236       "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]  
1237     },  
1238     {  
1239       "href": "/my/fan/1",  
1240       "rt": ["oic.r.switch.binary"],  
1241       "if": ["oic.if.a", "oic.if.baseline"],  
1242       "eps": [  
1243         {"ep": "coaps://[2001:db8:a::b1d4]:55555"}]  
1244         {  
1245           "href": "/his/fan/2",  
1246           "rt": ["oic.r.switch.binary"],  
1247           "if": ["oic.if.a", "oic.if.baseline"],  
1248           "eps": [  
1249             {"ep": "coaps://[2001:db8:a::b1d4]:55555"}]  
1250           }  
1251         ]  
1252     }  
1253 ]
```

### 1252 7.6.3.4 Batch OCF Interface

#### 1253 7.6.3.4.1 Overview

1254 The batch OCF Interface is used to interact with a Collection of Resources using a single/same  
1255 Request. The batch OCF Interface can be used to RETRIEVE or UPDATE the Properties of the  
1256 linked Resources with a single request.

1257 The batch OCF Interface is defined as follows:

- 1258 – The batch OCF Interface name is "oic.if.b"
- 1259 – A Collection Resource has linked Resources that are represented as URIs. In the "href"  
1260 Property of the batch payload the URI shall be fully qualified for remote Resources and a  
1261 relative reference for local Resources.
- 1262 – The original request is modified to create new requests targeting each of the linked Resources  
1263 in the Collection by substituting the URI in the original request with the URI of the linked  
1264 Resource. The payload in the original request is replicated in the payload of the new requests.
- 1265 – The requests shall be forwarded assuming use of the Default OCF Interface of the linked  
1266 Resources.
- 1267 – Requests shall only be forwarded to linked Resources that are identified by relation types "item"  
1268 or "hosts" ("hosts" is the default relation type value should the "rel" Link Parameter not be  
1269 present). Requests shall not be forwarded to linked Resources that do not contain the "item" or  
1270 "hosts" relation type values.

- 1271 – Properties of the Collection Resource itself may be included in payloads using "oic.if.b" OCF  
1272 Interface by exposing a single Link with the link relation "self" along with "item" within the  
1273 Collection, and ensuring that Link resolution cannot become an infinite loop due to recursive  
1274 references. For example, if the Default OCF Interface of the Collection is "oic.if.b", then the  
1275 Server might recursively include its batch representation within its batch representation, in an  
1276 endless loop. See 7.6.3.4.2 for an example of use of a Link containing "rel": ["self","item"] to  
1277 include Properties of the Collection Resource, along with linked Resources, in "oic.if.b"  
1278 payloads.
- 1279 – If the Default OCF Interface of a Collection Resource is exposed using the Link relation "self",  
1280 and the Default OCF Interface contains Properties that expose any Links, those Properties shall  
1281 not be included in a batch representation which includes the "self" Link.
- 1282 – Any request forwarded to a linked Resource that is a Collection (including a "self" Link reference)  
1283 shall have the Default OCF Interface of the linked Collection Resource applied.
- 1284 – All the responses from the linked Resources shall be aggregated into a single Response to the  
1285 Client. The Server may timeout the response to a time window, the Server may choose any  
1286 appropriate window based on conditions.
- 1287 – If a linked Resource cannot process the request, an empty response, i.e. a JSON object with  
1288 no content ("{}") as the representation for the "rep" Property, or error response should the linked  
1289 Resource Type provide an error schema or diagnostic payload, shall be returned by the linked  
1290 Resource. These empty or error responses for all linked Resources that exhibit an error shall  
1291 be included in the aggregated response to the original Client request. See the example in  
1292 7.6.3.4.2.
- 1293 – If any of the linked Resources returns an error response, the aggregated response sent to the  
1294 Client shall also indicate an error (e.g. 4.xx in CoAP). If all of the linked Resources return  
1295 successful responses, the aggregated response shall include the success response code.
- 1296 – The aggregated response shall be an array of objects representing the responses from each  
1297 linked Resource. Each object in the response shall include at least two items: (1) the URI of  
1298 the linked Resource (fully qualified for remote Resources, or a relative reference for local  
1299 Resources) as "href": <URI> and (2) the individual response object or array of objects if the  
1300 linked Resource is itself a Collection using "rep" as the key, e.g. "rep": { <representation of  
1301 individual response> }.
- 1302 – If the Collection Resource is marked as Observable, linked Resources referenced in the  
1303 Collection may be Observed using the batch OCF Interface. If the Collection Resource is not  
1304 marked as Observable then the Collection cannot be Observed and Observe requests to the  
1305 Collection shall be handled as defined for the case where request validation fails in clause  
1306 11.3.2.4. The Observe mechanism shall work as defined in 11.3.2 with the Observe request  
1307 forwarded to each of the linked Resources. All responses to the request shall be aggregated  
1308 into a single response to the Client using the same representations and status codes as for  
1309 RETRIEVE operations using the batch OCF Interface.
- 1310 – Should any one of the Observable linked Resources fail to honour the Observe request the  
1311 response to the batch Observe request shall also indicate that the entire request was not  
1312 honoured using the mechanism described in 11.3.2.4.
- 1313 – If any of the Observable Resources in a request to a Collection using the batch OCF Interface  
1314 replies with an error or Observe Cancel, the Observations of all other linked Resources shall  
1315 be cancelled and the error or Observe Cancel status shall be returned to the Observing Client.
- 1316 NOTE Behavior may be different for Links that do network requests vs. local Resources.
- 1317 – All notifications to the Client that initiated an Observe request using the batch OCF Interface  
1318 shall use the batch representation for the Collection. This is the aggregation of any individual  
1319 Observe notifications received by the Device hosting the Collection from the individual Observe  
1320 requests that were forwarded to the linked Resources.

- 1321 – Linked Resources which are not marked Observable in the Links of a Collection shall not trigger  
1322 Notifications, but may be included in the response to, and subsequent Notifications resulting  
1323 from, an Observe request to the batch OCF Interface of a Collection.
- 1324 – Each notification shall contain the most current values for all of the Linked Resources that would  
1325 be included if the original Observe request were processed again. The Server hosting the  
1326 Collection may choose to RETRIEVE all of the linked Resources each time, or may choose to  
1327 employ caching to avoid retrieving linked Resources on each Notification.
- 1328 – If a Linked Resource is Observable and has responded with a successful Observe response,  
1329 the most recently reported value of that Resource is considered to be the most current value  
1330 and may be reported in all subsequent Notifications.
- 1331 – Links in the Collection should be Observed by using the "oic.if.ll" OCF Interface. A notification  
1332 shall be sent any time the contents of the "oic.if.ll" OCF Interface representation are changed;  
1333 that is, if a Link is added, if a Link is removed, or if a Link is updated. Notifications on the  
1334 "oic.if.ll" OCF Interface shall contain all of the Links in the "oic.if.ll" OCF Interface representation.
- 1335 – Other Properties of the Collection Resource, if present, may be Observed by using the OCF  
1336 Interfaces defined in the definition for the Resource Type, including using the "oic.if.baseline"  
1337 OCF Interface.
- 1338 – The Client may choose to restrict the linked Resources to which the request is forwarded by  
1339 including additional query parameters in the request. The Server should process any additional  
1340 query parameters in a request that includes "oic.if.b" as selectors for linked Resources that are  
1341 to be processed by the request.
- 1342 – A Client shall perform UPDATE operations using the batch OCF Interface by creating a payload  
1343 that is similar to a RETRIEVE response payload from a batch OCF Interface request. The Server  
1344 shall send a separate UPDATE request to each of the linked Resources according to each "href"  
1345 Property and the corresponding value of the "rep" Property.
- 1346 – If the "href" value is empty, denoted by a zero length string or "" in JSON, the "rep" Property  
1347 shall be applied to linked Resources in the Collection.
- 1348 – Items with the empty "href" and link-specific "href" shall not be mixed in the same UPDATE  
1349 request.
- 1350 – All of the Properties in the UPDATE request may not be supported by the linked Resource. In  
1351 such cases, writable Properties in the UPDATE request that are supported by the linked  
1352 Resource shall be modified and Properties that are not supported shall be silently ignored.
- 1353 – The UPDATE response shall contain the updated values using the same payload schema as  
1354 RETRIEVE operations if provided by the linked Resource, along with the appropriate status  
1355 code. The aggregated response payload shall reflect the known state of the updated Properties  
1356 after the batch update was completed. If no payload is provided by the updated Resource then  
1357 an empty response (i.e. "rep": {}) shall be provided for that Resource.
- 1358 – A Collection shall not support the use of the UPDATE operation to add, modify, or remove Links  
1359 in an existing Collection using the "oic.if.baseline", "oic.if.rw" or "oic.if.a" OCF Interfaces.

#### 1360 **7.6.3.4.2 Examples: Batch OCF Interface**

1361 Note that the examples provided in Table 9 are illustrative and do not include all mandatory schema  
1362 elements in all cases. It is assumed that the Default OCF Interface for the Resource Type  
1363 "x.org.example.rt.room" is specified in its Resource Type definition file as "oic.if.rw", which exposes  
1364 the Properties "x.org.example.colour" and "x.org.example.size".

**Table 9 – Batch OCF Interface Example**

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



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

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

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

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

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

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

1366

### 1367 7.6.3.5 Actuator OCF Interface

1368 The actuator OCF Interface is the OCF Interface for viewing Resources that may be actuated i.e.  
1369 changes some value within or the state of the entity abstracted by the Resource:

- 1370 – The actuator OCF Interface name shall be "oic.if.a"
- 1371 – The actuator OCF Interface shall expose in the Resource Representation all mandatory  
1372 Properties as defined by the applicable OpenAPI 2.0 schema; the actuator OCF Interface may  
1373 also expose in the Resource Representation optional Properties as defined by the applicable  
1374 OpenAPI 2.0 schema that are implemented by the target Device.

1375 For example, a "Heater" Resource (for illustration only):

```
1376 /a/act/heater
1377 {
1378   "rt": ["acme.gas"],
1379   "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
1380   "settemp": 10,
1381   "currenttemp" : 7
1382 }
```

1383 The actuator OCF Interface with respect to "Heater" Resource (for illustration only):

1384

#### 1385 a) Retrieving values of an actuator.

1386 Request: GET /a/act/heater?if="oic.if.a"

1387

1388 Response:

```
1389 {
1390   "settemp": 10,
1391   "currenttemp" : 7
1392 }
```

#### 1393 b) Correct use of actuator OCF Interface.

1394

1395 Request: POST /a/act/heater?if="oic.if.a"

1396

```
1397 {
1398   "settemp": 20
1399 }
```

1399 Response:

```
1400 {
1401   Ok
1402 }
```

#### 1403 c) Incorrect use of actuator OCF Interface.

1404

1405 Request: POST /a/act/heater?if="oic.if.a"  
1406 {  
1407 "if": ["oic.if.s"] ← this is visible through baseline OCF Interface  
1408 }  
1409 Response:  
1410 {  
1411 Error  
1412 }

1413 – A RETRIEVE request using this OCF Interface shall return the Representation for this Resource  
1414 subject to any query and filter parameters that may also exist.

1415 – An UPDATE request using this OCF Interface shall provide a payload or body that contains the  
1416 Properties that will be updated on the target Resource.

1417 **7.6.3.6 Sensor OCF Interface**

1418 The sensor OCF Interface is the OCF Interface for retrieving measured, sensed or capability  
1419 specific information from a Resource that senses:

1420 – The sensor OCF Interface name shall be "oic.if.s".

1421 – The sensor OCF Interface shall expose in the Resource Representation all mandatory  
1422 Properties as defined by the applicable OpenAPI 2.0 schema; the sensor OCF Interface may  
1423 also expose in the Resource Representation optional Properties as defined by the applicable  
1424 OpenAPI 2.0 schema that are implemented by the target Device.

1425 – A RETRIEVE request using this OCF Interface shall return this representation for the Resource  
1426 subject to any query and filter parameters that may also exist.

1427 NOTE: The example here is with respect to retrieving values of a sensor

1428  
1429 Request: GET /a/act/heater?if="oic.if.s"  
1430  
1431 Response:  
1432 {  
1433 "currenttemp": 7  
1434 }  
1435

1436 **Incorrect use of the sensor.**

1437 Request: PUT /a/act/heater?if="oic.if.s" ← PUT is not allowed  
1438 {  
1439 "settemp": 20 ← this is possible through actuator OCF Interface  
1440 }  
1441 Response:  
1442 {  
1443 Error  
1444 }  
1445

1446 **Another incorrect use of the sensor.**

1447 Request: POST /a/act/heater?if="oic.if.s" ← POST is not allowed  
1448 {  
1449 "currenttemp": 15 ← this is possible through actuator OCF Interface  
1450 }  
1451 Response:  
1452 {  
1453 Error  
1454 }

1455 **7.6.3.7 Read-only OCF Interface**

1456 The read-only OCF Interface exposes only the Properties that may be read. This includes  
1457 Properties that may be read-only, read-write but not Properties that are write-only or set-only. The  
1458 applicable operations that can be applied to a Resource are only RETRIEVE and NOTIFY. An  
1459 attempt by a Client to apply a method other than RETRIEVE or NOTIFY to a Resource shall be  
1460 rejected with an error response code.

1461 **7.6.3.8 Read-write OCF Interface**

1462 The read-write OCF Interface is a generic OCF Interface to support reading and setting Properties  
1463 in a Resource. The applicable methods that can be applied to a Resource are only RETRIEVE,  
1464 NOTIFY, and UPDATE. For the RETRIEVE and NOTIFY operations, the behaviour is the same as  
1465 for the "oic.if.r" OCF Interface defined in 7.6.3.7. For the UPDATE operation, read-only Properties  
1466 (i.e. Properties tagged with "readOnly=True" in the OpenAPI 2.0 definition) shall not be in the  
1467 UPDATE payload. An attempt by a Client to apply a method other than RETRIEVE, NOTIFY, or  
1468 UPDATE to a Resource shall be rejected with an error response code.

1469 **7.6.3.9 Create OCF Interface**

1470 **7.6.3.9.1 Overview**

1471 The create OCF Interface is used to create Resource instances in a Collection. An instance of a  
1472 Resource and the Link pointing to the Resource are created together, atomically, according to a  
1473 Client-supplied representation. The create OCF Interface name is "oic.if.create". A Collection which  
1474 exposes the "oic.if.create" OCF Interface shall expose the "rts" Property (see clause 7.8.2.5) with  
1475 all Resource Types that can be hosted with the Collection. If a Client attempts to create a Resource  
1476 Type which is not supported by the Collection, the Server shall return an appropriate error status  
1477 code, for example "Bad Request". Successful CREATE operations shall return a success code, i.e.  
1478 "Created". The IDD for all allowed Resource Types that may be created shall adhere to  
1479 Introspection for dynamic Resources (see clause 11.4).

1480 **7.6.3.9.2 Data format for CREATE**

1481 The data format for the create OCF Interface is similar to the data format for the batch OCF  
1482 Interface. The create OCF Interface format consists of a set of Link Parameters and a "rep"  
1483 Parameter which contains a representation for the created Resource.

1484 The representation supplied for the Link pointing to the newly created Resource shall contain at  
1485 least the "rt" and "if" Link Parameters.

1486 The Link Parameter "p" should be included in representations supplied for all created Resources.  
1487 If the "Discoverable" bit is set, then the supplied Link representation shall be exposed in "/oic/res"  
1488 of the Device on which the Resource is being created. The Link Parameters representation in the  
1489 "/oic/res" Resource does not have to mirror the Link Parameters in the Collection of the created  
1490 Resource (e.g., "ins" Parameter).

1491 Creating a discoverable Resource is the only way to add a Link to "/oic/res".

1492 If the "p" Parameter is not included, the Server shall create the Resource using the default settings  
1493 of not discoverable, and not observable.

1494 The representation supplied for a created Resource in the value of the "rep" Parameter shall  
1495 contain all mandatory Properties required by the Resource Type to be created excluding the  
1496 Common Properties "rt" and "if" as they are already included in the create payload.

1497 Note that the "rt" and "if" Property Values are created from the supplied Link Parameters of the  
1498 Resource creation payload.



1499 If the supplied representation does not contain all of the required Properties and Link Parameters,  
1500 the Server shall return an appropriate error status code, for example "Bad Request".

1501 An example of the create OCF Interface payload is as illustrated:

```
1502 {  
1503   "rt": ["oic.r.temperature"],  
1504   "if": ["oic.if.a","oic.if.baseline"],  
1505   "p": {"bm":3},  
1506   "rep": {  
1507     "temperature": 20  
1508   }  
1509 }
```

1510 The representation returned when a Resource is successfully created shall contain the "href", "if",  
1511 and "rt" Link Parameters and all other Link Parameters that were included in the CREATE operation.  
1512 In addition, the "rep" Link Parameter shall include all Resource Properties as well as the "rt" and  
1513 "if" Link Parameters supplied in the CREATE operation. The Server may include additional Link  
1514 Parameters and Properties in the created Resource as required by the application-specific  
1515 Resource Type. The Server shall assign an "ins" value to each created Link and shall include the  
1516 "ins" Parameter in the representation of each created Link as illustrated in the Collection that the  
1517 Link of the created Resource was created within:

```
1518 {  
1519   "href": "/3755f3ac",  
1520   "rt": ["oic.r.temperature"],  
1521   "if": ["oic.if.a","oic.if.baseline"],  
1522   "ins": 39724818,  
1523   "p": {"bm":3},  
1524   "rep": {  
1525     "rt": ["oic.r.temperature"],  
1526     "if": ["oic.if.a","oic.if.baseline"],  
1527     "temperature": 20  
1528   }  
1529 }
```

1530 The Link Parameters representation in the "/oic/res" Resource, if the created Resource is  
1531 discoverable, may not mirror exactly all the Link Parameters added in the Collection; except it shall  
1532 expose at a minimum the mandatory Properties of the Link (i.e., "rt", "if", and "href") of the created  
1533 Resource.

### 1534 7.6.3.9.3 Use with CREATE

1535 The CREATE operation shall be sent to the URI of the Collection in which the Resource is to be  
1536 created. The query string "?if=oic.if.create" shall be included in all CREATE operations.

1537 The Server shall generate a URI for the created Resource and include the URI in the "href"  
1538 Parameter of the created Link.

1539 When a Server successfully completes a CREATE operation using the "oic.if.create" OCF Interface  
1540 addressing a Collection, the Server shall automatically modify the ACL Resource to provide initial  
1541 authorizations for accessing for the newly created Resource according to ISO/IEC 30118-2:2018.

1542 An example performing a CREATE operation is as illustrated:

```
1543 CREATE /scenes/scenel?if=oic.if.create  
1544 {  
1545   "rt": ["oic.r.temperature"],  
1546   "if": ["oic.if.a","oic.if.baseline"],  
1547   "p": {"bm":3},  
1548   "rep": {  
1549     "temperature": 20
```

```
1550     }
1551   }
1552   Response: Created
1553   Payload:
1554   {
1555     "href": "/3755f3ac",
1556     "ins": 39724818,
1557     "rt": ["oic.r.temperature"],
1558     "if": ["oic.if.a","oic.if.baseline"],
1559     "p": {"bm":3},
1560     "rep": {
1561       "rt": ["oic.r.temperature"],
1562       "if": ["oic.if.a","oic.if.baseline"],
1563       "temperature": 20
1564     }
1565   }
```

#### 1566 **7.6.3.9.4 Use with UPDATE and DELETE**

1567 The UPDATE and DELETE operations are not allowed by the create OCF Interface. Attempts to  
1568 perform UPDATE or DELETE operations using the create OCF Interface shall return an appropriate  
1569 error status code, for example "Method Not Allowed", unless the UPDATE and CREATE operations  
1570 map to the same transport binding method (e.g., CoAP with the POST method). In that situation  
1571 where the UPDATE and CREATE operations map to the same transport binding method, this shall  
1572 be processed as a CREATE operation according to clause 7.6.3.9.3.

### 1573 **7.7 Resource representation**

1574 Resource representation captures the state of a Resource at a particular time. The Resource  
1575 representation is exchanged in the request and response interactions with a Resource. A Resource  
1576 representation may be used to retrieve or update the state of a Resource.

1577 The Resource representation shall not be manipulated by the data connectivity protocols and  
1578 technologies (e.g., CoAP, UDP/IP or BLE).

### 1579 **7.8 Structure**

#### 1580 **7.8.1 Introduction**

1581 In many scenarios and contexts, the Resources may have either an implicit or explicit structure  
1582 between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The  
1583 Framework provides the means to model and map these structures and the relationships among  
1584 Resources. The primary building block for Resource structures in Framework is the Collection. A  
1585 Collection represents a container, which is extensible to model complex structures.

#### 1586 **7.8.2 Resource Relationships**

##### 1587 **7.8.2.1 Introduction**

1588 Resource relationships are expressed as Links. A Link embraces and extends typed web links  
1589 concept as a means of expressing relationships between Resources. A Link consists of a set of  
1590 Parameters that define:

- 1591 – a context URI,
- 1592 – a target URI,
- 1593 – a relation from the context URI to the target URI, and
- 1594 – elements that provide metadata about the target URI, the relationship or the context of the Link.

1595 The target URI is mandatory and the other items in a Link are optional. Additional items in the Link  
1596 may be made mandatory based on the use of the links in different contexts (e.g. in Collections, in  
1597 discovery, in bridging etc.). OpenAPI 2.0 schema for the Link payload is provided in Annex A.

1598 An example of a Link is:

```
1599 {"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a",  
1600 "oic.if.baseline"], "p": {"bm": 3}, "rel": "item"}
```

1601 Two Links are distinct from each other when at least one Parameter is different. For example the  
1602 two Links show here are distinct and can appear in the same list of Links.

```
1603 {"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a",  
1604 "oic.if.baseline"], "p": {"bm": 2}, "rel": "item"}  
1605 {"href": "/switch", "rt": ["oic.r.switch.binary"], "if": ["oic.if.a",  
1606 "oic.if.baseline"], "p": {"bm": 2}}
```

1607 The document may mandate Parameters and Parameter values as required for certain capabilities.  
1608 For all Links returned in a response to a RETRIEVE on "/oic/res", if a Link does not explicitly include  
1609 the "rel" Parameter, a value of "rel"="hosts" shall be assumed. The relation value of "hosts" is  
1610 defined by IETF RFC 6690, the value of "item" by IETF RFC 6573, and the value of "self" by  
1611 IETF RFC 4287 and all are registered in the IANA Registry for Link Relations defined in  
1612 IANA Link Relations.

1613 As shown in Annex A the relation between the context URI and target URI in a Link is specified  
1614 using the "rel" JSON element and the value of this element specifies the particular relation.

1615 The context URI of the Link shall implicitly be the URI of the Resource (or specifically a Collection)  
1616 that contains the Link unless the Link specifies the "anchor" Parameter. The "anchor" Parameter  
1617 is used to change the context URI of a Link – the relationship with the target URI is based off the  
1618 anchor URI when the "anchor" is specified. "Anchor" Parameter uses transfer protocol URI for OIC  
1619 1.1 Link (e.g. "anchor": "coaps://[fe80::b1d6]:44444") and OCF URI defined in Sec 6 for OCF 1.0  
1620 Links (e.g. "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989"). For optional backward  
1621 compatibility with OIC 1.1, "anchor" Parameter uses transfer protocol URI for OIC 1.1 Link (e.g.  
1622 "anchor": "coaps://[fe80::b1d6]:44444").

1623 An example of using "anchors" in the context of Collections – a floor has rooms and rooms have  
1624 lights – the lights may be defined in floor as Links but the Links will have the "anchor" set to the  
1625 URI of the rooms that contain the lights (the relation is contains). This allows all lights in a floor to  
1626 be turned on or off together while still having the lights defined with respect to the rooms that  
1627 contain them (lights may also be turned on by using the room URI too). For example, here is the  
1628 use of "anchor" in Link:

```
1629 /a/floor {  
1630   "links": [  
1631     {  
1632       "href": "/x/light1",  
1633       "anchor": "/a/room1",    ** Note: /a/room1 has the item relationship with /x/light1; not /a/floor **  
1634       "rel": "item"  
1635     }  
1636   ]  
1637 }  
1638  
1639 /a/room1 {  
1640   "links": [  
1641     {  
1642       ** Note: /a/room1 "contains" the /x/light since /a/room1 is the implicit context URI **  
1643       "href": "/x/light1",  
1644       "rel": "item"  
1645     }  
1646   ]  
1647 }
```

1648 **7.8.2.2 Parameters**

1649 **7.8.2.2.1 "ins" or Link Instance Parameter**

1650 The "ins" Parameter identifies a particular Link instance in a list of Links. The "ins" Parameter may  
 1651 be used to modify or delete a specific Link in a list of Links. The value of the "ins" Parameter is set  
 1652 at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has  
 1653 been set, the "ins" Parameter shall not be modified for as long as the Link is a member of that list.

1654 **7.8.2.2.2 "p" or Policy Parameter**

1655 The Policy Parameter defines various rules for correctly accessing a Resource referenced by a  
 1656 target URI. The Policy rules are configured by a set of key-value pairs.

1657 The policy Parameter "p" is defined by:

- 1658 – "bm" key: The "bm" key corresponds to an integer value that is interpreted as an 8-bit bitmask.  
 1659 Each bit in the bitmask corresponds to a specific Policy rule. The rules are specified for "bm"  
 1660 in Table 10:

1661 **Table 10 – "bm" Property definition**

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

1662  
 1663 NOTE If all the bits in "bm" are defined to value 0, then the "bm" key may be omitted entirely from "p" as an efficiency  
 1664 measure. However, if any bit is set to value 1, then "bm" shall be included in "p" and all the bits shall be defined  
 1665 appropriately.

- 1666 – In a payload sent in response to a request that includes an OCF-Accept-Content-Format-  
 1667 Version option the "eps" Parameter shall provide the information for an encrypted connection.
- 1668 – Note that access to the Resource is controlled by the ACL for the Resource. A successful  
 1669 encrypted connection does not ensure that the requested action will succeed. See  
 1670 ISO/IEC 30118-2:2018 clause 12 for more information.

1671 This shows the Policy Parameter for a Resource that is discoverable but not Observable.

1672 "p": {"bm": 1}

1673 This shows a self-link, i.e. the "/oic/res" Link in itself that is discoverable and Observable.

```
1674 {
1675   "href": "/oic/res",
1676   "rel": "self",
1677   "rt": ["oic.wk.res"],
1678   "if": ["oic.if.ll", "oic.if.baseline"],
1679   "p": {"bm": 3}
1680 }
```

#### 1681 **7.8.2.2.3 "type" or Media Type Parameter**

1682 The "type" Parameter may be used to specify the various media types that are supported by a  
1683 specific target Resource. The default type of "application/vnd.ocf+cbor" shall be used when the  
1684 "type" element is omitted. Once a Client discovers this information for each Resource, it may use  
1685 one of the available representations in the appropriate header field of the Request or Response.

#### 1686 **7.8.2.2.4 "di" or Device ID Parameter**

1687 The "di" Parameter specifies the Device ID of the Device that hosts the target Resource defined in  
1688 the in the "href" Parameter.

1689 The Device ID may be used to qualify a relative reference used in the "href" or to lookup OCF  
1690 Endpoint information for the relative reference.

#### 1691 **7.8.2.2.5 "eps" Parameter**

1692 The "eps" Parameter indicates the OCF Endpoint information of the target Resource.

1693 "eps" shall have as its value an array of items and each item represents OCF Endpoint information  
1694 with "ep" and "pri" as specified in 10.2. "ep" is mandatory but "pri" is optional.

1695 This is an example of "eps" with multiple OCF Endpoints.

```
1696 "eps": [
1697   {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
1698   {"ep": "coaps://[fe80::b1d6]:1122"},
1699   {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
1700 ]
```

1701 When "eps" is present in a link, the OCF Endpoint information in "eps" can be used to access the  
1702 target Resource referred by the "href" Parameter.

1703 Note that the type of OCF Endpoint – Secure or Unsecure – that a Resource exposes merely  
1704 determines the connection type(s) guaranteed to be available for sending requests to the Resource.  
1705 For example, if a Resource only exposes a single CoAP "ep", it does not guarantee that the  
1706 Resource cannot also be accessed via a Secure OCF Endpoint (e.g. via a CoAPS "ep" from another  
1707 Resource's "eps" information). Nor does exposing a given type of OCF Endpoint ensure that access  
1708 to the Resource will be granted using the "ep" information. Whether requests to the Resource are  
1709 granted or denied by the Access Control layer is separate from the "eps" information, and is  
1710 determined by the configuration of the /acl2 Resource (see ISO/IEC 30118-2:2018 clause 13.5.3  
1711 for details).

1712 When present, max-age information (e.g. Max-Age option for CoAP defined in IETF RFC 7252)  
1713 determines the maximum time "eps" values may be cached before they are considered stale.

#### 1714 **7.8.2.3 Formatting**

1715 When formatting in JSON, the list of Links shall be an array.

1716 **7.8.2.4 List of Links in a Collection**

1717 A Resource that exposes one or more Properties that are defined to be an array of Links where  
 1718 each Link can be discretely accessed is a Collection. The Property Name "links" is recommended  
 1719 for such an array of Links.

1720 This is an example of a Resource with a list of Links.

```

1721 /Room1
1722 {
1723   "rt": ["oic.wk.col"],
1724   "if": ["oic.if.ll", "oic.if.baseline" ],
1725   "color": "blue",
1726   "links":
1727   [
1728     {
1729       "href": "/switch",
1730       "rt": ["oic.r.switch.binary"],
1731       "if": [ "oic.if.a", "oic.if.baseline" ],
1732       "p": {"bm": 3}
1733     },
1734     {
1735       "href": "/brightness",
1736       "rt": ["oic.r.light.brightness"],
1737       "if": [ "oic.if.a", "oic.if.baseline" ],
1738       "p": {"bm": 3}
1739     }
1740   ]
1741 }
  
```

1742 **7.8.2.5 Properties describing an array of Links**

1743 If a Resource Type that defines an array of Links (e.g. Collections, Atomic Measurements) has  
 1744 restrictions on the "rt" values that can be within the array of Links, the Resource Type will define  
 1745 the "rts" Property. The "rts" Property as defined in Table 11 will include all "rt" values allowed for  
 1746 all Links in the array. If the Resource Type does not define the "rts" Property or the "rts" Property  
 1747 is an empty array, then any "rt" value is permitted in the array of Links.

1748 For all instances of a Resource Type that defines the "rts" Property, the "rt" Link Parameter in  
 1749 every Link in the array of Links shall be one of the "rt" values that is included in the "rts"  
 1750 Property.

1751 **Table 11 – Resource Types Property definition**

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

1752

1753 If a Resource Type that defines an array of Links has "rt" values which are required to be in the  
 1754 array, the Resource Type will define the "rts-m" Property, as defined in Table 12, which will contain  
 1755 all of the "rt" vaues that are required to be in the array of Links. If "rts-m" is defined, and "rts" is  
 1756 defined and is not an empty array, then the "rt" values present in "rts-m" will be part of the values  
 1757 present in "rts". Moreover, if the "rts-m" Property is defined, it shall be mandated (i.e. included in  
 1758 the "required" field of a JSON definition) in the Resource definition and Introspection Device Data  
 1759 (see 11.4).

1760 For all instances of a Resource Type that defines the "rts-m" Property, there shall be at least one  
 1761 Link in the array of Links corresponding to each one of the "rt" values in the "rts-m" Property; for  
 1762 all such Links the "rt" Link Parameter shall contain at least one of the "rt" values in the "rts-m"  
 1763 Property.

1764 **Table 12 – Mandatory Resource Types Property definition**

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

1765

1766 **7.8.3 Collections**

1767 **7.8.3.1 Overview**

1768 A Resource that contains one or more references (specified as Links) to other Resources is a  
 1769 Collection. These references may be related to each other or just be a list; the Collection provides  
 1770 a means to refer to this set of references with a single handle (i.e. the URI). A simple Resource is  
 1771 kept distinct from a Collection. Any Resource may be turned into a Collection by binding Resource  
 1772 references as Links. Collections may be used for creating, defining or specifying hierarchies,  
 1773 indexes, groups, and so on.

1774 A Collection shall have at least one Resource Type and at least one OCF Interface bound at all  
 1775 times during its lifetime. During creation time of a Collection the Resource Type and OCF Interfaces  
 1776 are specified. The initial defined Resource Types and OCF Interfaces may be updated during its  
 1777 life time. These initial values may be overridden using mechanism used for overriding in the case  
 1778 of a Resource. Additional Resource Types and OCF Interfaces may be bound to the Collection at  
 1779 creation or later during the lifecycle of the Collection.

1780 A Collection shall define a Property that is an array with zero or more Links. The target URIs in the  
 1781 Links may reference another Collection or another Resource. The referenced Collection or  
 1782 Resource may reside on the same Device as the Collection that includes that Link (called a local  
 1783 reference) or may reside on another Device (called a remote reference). The context URI of the  
 1784 Links in the array shall (implicitly) be the Collection that contains that Property. The (implicit)  
 1785 context URI may be overridden with explicit specification of the "anchor" Parameter in the Link  
 1786 where the value of "anchor" is the new base of the Link.

1787 A Resource may be referenced in more than one Collection, therefore, a unique parent-child  
 1788 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the  
 1789 Resource referenced in the Collection, i.e., the application may use Collections to represent a  
 1790 relationship but none is automatically implied or defined. The lifecycles of the Collection and the  
 1791 referenced Resource are also independent of one another.

1792 If the "drel" Property is defined for the Collection then all Links that don't explicitly specify a  
 1793 relationship shall inherit this default relationship in the context of that Collection. The default  
 1794 relationship defines the implicit relationship between the Collection and the target URI in the Link.

1795 In the following example a Property "links" represents the list of Links in a Collection. The "links"  
 1796 Property has, as its value, an array of items and each item is a Link.

```
1797 /my/house ← This is IRI/URI of the Resource
1798 {
```

```

1799     "rt": ["my.r.house"],    ← This and the next 3 lines are the Properties of the
1800 Resource.
1801     "color": "blue",
1802     "n": "myhouse",
1803     "links": [
1804       {    ← This and the next 4 lines are the Parameters of a Link
1805         "href": "/door",
1806         "rt": ["oic.r.door"],
1807         "if": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
1808       },
1809
1810       {
1811         "href": "/door/lock",
1812         "rt": ["oic.r.lock"],
1813         "if": ["oic.if.b", "oic.if.baseline"]
1814       },
1815
1816       {
1817         "href": "/light",
1818         "rt": ["oic.r.light"],
1819         "if": ["oic.if.s", "oic.if.baseline"]
1820       },
1821
1822       {
1823         "href": "/binarySwitch",
1824         "rt": ["oic.r.switch.binary"],
1825         "if": ["oic.if.a", "oic.if.baseline"]
1826       }
1827     ]
1828 }
1829 }

```

1830 A Collection may be:

- 1831 – A pre-defined Collection where the Collection has been defined a priori and the Collection is
- 1832 static over its lifetime. Such Collections may be used to model, for example, an appliance that
- 1833 is composed of other Devices or fixed set of Resources representing fixed functions.
- 1834 – A Device local Collection where the Collection is used only on the Device that hosts the
- 1835 Collection. Such Collections may be used as a short-hand on a Client for referring to many
- 1836 Servers as one.
- 1837 – A centralized Collection where the Collection is hosted on a Device but other Devices may
- 1838 access or update the Collection.
- 1839 – A hosted Collection where the Collection is centralized but is managed by an authorized agent
- 1840 or party.

### 1841 7.8.3.2 Collection Properties

1842 A Collection shall define a Property that is an array of Links (the Property Name "links" is

1843 recommended). In addition, other Properties may be defined for the Collection by the Resource

1844 Type. The mandatory and recommended Common Properties for a Collection are shown in Table 13.

1845 This list of Common Properties is in addition to those defined for Resources in 7.3.2.

1846 **Table 13 – Common Properties for Collections (in addition to Common Properties defined**

1847 **in 7.3.2)**

Property	Description	Property Name	Value Type	Mandatory
Links	The array of Links in the Collection	Per Resource Type definition	json Array of Links	Yes



<b>Resource Types</b>	The list of allowed Resource Types for Links in the Collection. If this Property is not defined or is null string then any Resource Type is permitted	As defined in Table 11	As defined in Table 11	No
<b>Mandatory Resource Types</b>	The list of Resource Types for Links that are mandatory in the Collection.	As defined in Table 12	As defined in Table 12	No

1848

1849 **7.8.3.3 Default Resource Type**

1850 A default Resource Type, "oic.wk.col", is available for Collections. This Resource Type shall be  
 1851 used only when another type has not been defined on the Collection or when no Resource Type  
 1852 has been specified at the creation of the Collection.

1853 The default Resource Type provides support for the Common Properties including an array of Links  
 1854 with the Property Name "links".

1855 **7.8.3.4 Default OCF Interface**

1856 All instances of a Collection shall support the links list ("oic.if.ll") OCF Interface in addition to the  
 1857 baseline ("oic.if.baseline") OCF Interface. An instance of a Collection may optionally support  
 1858 additional OCF Interfaces that are defined within this document. The Default OCF Interface for a  
 1859 Collection shall be links list ("oic.if.ll") unless otherwise specified by the Resource Type definition.

1860 **7.8.4 Atomic Measurement**

1861 **7.8.4.1 Overview**

1862 Certain use cases require that the Properties of multiple Resources are only accessible as a group  
 1863 and individual access to those Properties of each Resource by a Client is prohibited. The Atomic  
 1864 Measurement Resource Type is defined to meet this requirement. This is accomplished through  
 1865 the use of the Batch OCF Interface.

1866 **7.8.4.2 Atomic Measurement Properties**

1867 An Atomic Measurement shall define a Property that is an array of Links (the Property Name "links"  
 1868 is recommended). In addition, other Properties may be defined for the Atomic Measurement by the  
 1869 Resource Type. The mandatory and recommended Common Properties for an Atomic  
 1870 Measurement are shown in Table 14. This list of Common Properties is in addition to those defined  
 1871 for Resources in 7.3.2.

1872 **Table 14 – Common Properties for Atomic Measurement (in addition to Common Properties**  
 1873 **defined in 7.3.2)**

Property	Description	Property Name	Value Type	Mandatory
<b>Links</b>	The array of Links in the Atomic Measurement	Per Resource Type definition	json Array of Links	Yes
<b>Resource Types</b>	The list of allowed Resource Types for Links in the Atomic Measurement. If this Property is not defined or is null	As defined in Table 11	As defined in Table 11	No

	string then any Resource Type is permitted			
<b>Mandatory Resource Types</b>	The list of Resource Types for Links that are mandatory in the Atomic Measurement.	As defined in Table 12	As defined in Table 12	No

1874

### 1875 **7.8.4.3 Normative behaviour**

1876 The normative behaviour of an Atomic Measurement is as follows:

- 1877 – The behaviour of the Batch OCF Interface ("oic.if.b") on the Atomic Measurement is defined as  
1878 follows:
  - 1879 – Only RETRIEVE and NOTIFY operations are supported, for Batch OCF Interface, on Atomic  
1880 Measurement; the behavior of the RETRIEVE and NOTIFY operations shall be the same as  
1881 specified in 7.6.3.4, with exceptions as provided for in 7.8.4.3.
  - 1882 – The UPDATE operation is not allowed, for Batch OCF Interface, on Atomic Measurement; if  
1883 an UPDATE operation is received, it shall result in a method not allowed error code.
  - 1884 – An error response shall not include any representation of a linked Resource (i.e. empty  
1885 response for all linked Resources).
- 1886 – Any linked Resource within an Atomic Measurement (i.e. the target Resource of a Link in an  
1887 Atomic Measurement) is subject to the following conditions:
  - 1888 – Linked Resources within an Atomic Measurement and the Atomic Measurement itself shall  
1889 exist on a single Server.
  - 1890 – CRUDN operations shall not be allowed on linked Resources and shall result in a forbidden  
1891 error code.
  - 1892 – Linked Resources shall not expose the "oic.if.ll" OCF Interface. Since CRUDN operations  
1893 are not allowed on linked Resources, the "oic.if.ll" OCF Interface would never be accessible.
- 1894 – Links to linked Resources in an Atomic Measurement shall only be accessible through the  
1895 "oic.if.ll" or the "oic.if.baseline" OCF Interfaces of an Atomic Measurement.
  - 1896 – The linked Resources shall not be listed in "/oic/res".
- 1897 – A linked Resource in an Atomic Measurement shall have defined one of "oic.if.a", "oic.if.s",  
1898 "oic.if.r", or "oic.if.rw" as its Default OCF Interface.
- 1899 – Not all linked Resources in an Atomic Measurement are required to be Observable. If an Atomic  
1900 Measurement is being Observed using the "oic.if.b" OCF Interface, notification responses shall  
1901 not be generated when the linked Resources which are not marked Observable are updated or  
1902 change state.
- 1903 – All linked Resources in an Atomic Measurement shall be included in every RETRIEVE and  
1904 Observe response when using the "oic.if.b" OCF Interface.
- 1905 – An Atomic Measurement shall support the "oic.if.b" and the "oic.if.ll" OCF Interfaces.
- 1906 – Filtering of linked Resources in an Atomic Measurement is not allowed. Query parameters that  
1907 select one or more individual linked Resources in a request to an Atomic Measurement shall  
1908 result in a "forbidden" error code.
- 1909 – If the "rel" Link Parameter is included in a Link contained in an Atomic Measurement, it shall  
1910 have either the "hosts" or the "item" value.
- 1911 – The Default OCF Interface of an Atomic Measurement is "oic.if.b".

1912 **7.8.4.4 Security considerations**

1913 Access rights to an Atomic Measurement Resource Type is as specified in clause 12.2.7.2 (ACL  
1914 considerations for batch request to the Atomic Measurement Resource Type) of ISO/IEC 30118-  
1915 2:2018).

1916 **7.8.4.5 Default Resource Type**

1917 The Resource Type is defined as "oic.wk.atomicmeasurement" as defined in Table 15.

1918 **Table 15 – Atomic Measurement Resource Type**

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

1919

1920 The Properties for Atomic Measurement are as defined in Table 16.

1921 **Table 16 – Properties for Atomic Measurement (in addition to Common Properties defined  
1922 in 7.3.2)**

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

1923

1924 **7.9 Query Parameters**

1925 **7.9.1 Introduction**

1926 Properties and Parameters (including those that are part of a Link) may be used in the query part  
1927 of a URI (see 6.2.2) as one criterion for selection of a particular Resource. This is done by declaring  
1928 the Property (i.e. <Property Name> = <desired Property Value>) as one of the segments of the  
1929 query. Only ASCII strings are permitted in query filters, and NULL characters are disallowed in  
1930 query filters. This means that only Property Values with ASCII characters may be matched in a  
1931 query filter.

1932 The Resource is selected when all the declared Properties or Link Parameters in the query match  
1933 the corresponding Properties or Link Parameters in the target.

1934 **7.9.2 Use of multiple parameters within a query**

1935 When a query contains multiple separate query parameters these are delimited by an "&" as  
1936 described in 6.2.2.

1937 A Client may apply multiple separate query parameters, for  
1938 example "?ins=11111&rt=oic.r.switch.binary". If such queries are supported by the Server this shall  
1939 be accomplished by matching "all of" the different query parameter types ("rt", "ins", "if", etc)  
1940 against the target of the query. In the example, this resolves to an instance of oic.r.switch.binary  
1941 that also has an "ins" populated as "11111". There is no significance applied to the order of the  
1942 query parameters.

1943 A Client may select more than one Resource Type using repeated query parameters, for example  
1944 "?rt=oic.r.switch.binary&rt=oic.r.ramptime". If such queries are supported by the Server this shall  
1945 be accomplished by matching "any of" the repeated query parameters against the target of the  
1946 query. In the example, any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that may exist  
1947 are selected.

1948 A Client may combine both multiple repeated parameters and multiple separate parameters in a  
1949 single query, for example "?if=oic.if.b&ins=11111&rt=oic.r.switch.binary&rt=oic.r.ramptime". If  
1950 such queries are supported by the Server this shall be accomplished by matching "any of" the  
1951 repeated query parameters and then matching "all of" the different query parameter types. In the  
1952 example any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that also have an "ins" of  
1953 "11111" that may exist are selected in a batch response.

1954 NOTE The parameters within a query string are represented within the actual messaging protocol as defined in clause  
1955 12.

### 1956 **7.9.3 Application to multi-value "rt" Resources**

1957 An "rt" query for a multi-value "rt" Resource with the Default OCF Interface of "oic.if.a", "oic.if.s",  
1958 "oic.if.r", "oic.if.rw" or "oic.if.baseline" is an extension of a generic "rt" query. When a Server  
1959 receives a RETRIEVE request for a multi-value "rt" Resource with an "rt" query, (i.e. GET  
1960 /ResExample?rt=oic.r.foo), the Server should respond only when the query value is an item of the  
1961 "rt" Property Value of the target Resource and should send back only the Properties associated  
1962 with the query value(s). For example, upon receiving GET /ResExample?rt=oic.r.switch.binary  
1963 targeting a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"], the Server responds  
1964 with only the Properties of oic.r.switch.binary.

### 1965 **7.9.4 OCF Interface specific considerations for queries**

#### 1966 **7.9.4.1 OCF Interface selection**

1967 When an OCF Interface is to be selected for a request, it shall be specified as a query parameter  
1968 in the URI of the Resource in the request message. If no query parameter is specified, then the  
1969 Default OCF Interface shall be used. If the selected OCF Interface is not one of the permitted OCF  
1970 Interfaces on the Resource then selecting that OCF Interface is an error and the Server shall  
1971 respond with an error response code.

1972 For example, the baseline OCF Interface may be selected by adding "if=oic.if.baseline" to the list  
1973 of query parameters in the URI of the target Resource. For example: "GET  
1974 /oic/res?if=oic.if.baseline".

#### 1975 **7.9.4.2 Batch OCF Interface**

1976 See 7.6.3.4 for details on the batch OCF Interface itself. Query parameters may be used with the  
1977 batch OCF Interface in order to select particular Resources in a Collection for retrieval or update;  
1978 these parameters are used to select items in the Collection by matching Link Parameter Values.

1979 When Link selection query parameters are used with RETRIEVE operations applied using the batch  
1980 OCF Interface, only the Resources in the Collection with matching Link Parameters should be  
1981 returned.

1982 When Link selection query parameters are used with UPDATE operations applied using the batch  
1983 OCF Interface, only the Resources having matching Link Parameters should be updated.

1984 See 7.6.3.4.2 for examples of RETRIEVE and UPDATE operations that use Link selection query  
1985 parameters.

1986 **8 CRUDN**

1987 **8.1 Overview**

1988 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for  
 1989 manipulating Resources. These operations are performed by a Client on the Resources contained  
 1990 in n Server.

1991 On reception of a valid CRUDN operation a Server hosting the Resource that is the target of the  
 1992 request shall generate a response depending on the OCF Interface included in the request; or  
 1993 based on the Default OCF Interface for the Resource Type if no OCF Interface is included.

1994 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in  
 1995 Table 17. A Device shall use CBOR as the default payload (content) encoding scheme for Resource  
 1996 representations included in CRUDN operations and operation responses; a Device may negotiate  
 1997 a different payload encoding scheme (e.g, see in 12.2.4 for CoAP messaging). Clauses 8.2 through  
 1998 8.6 respectively specify the CRUDN operations and use of the parameters. The type definitions for  
 1999 these terms will be mapped in the clause 12 for each protocol.

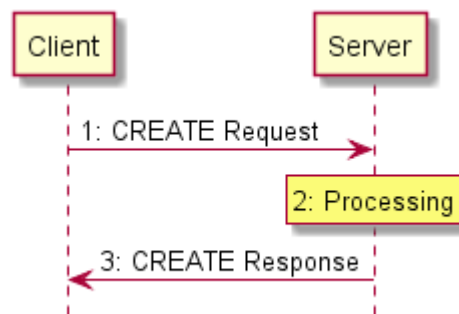
2000 **Table 17 – Parameters of CRUDN messages**

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

2001 **8.2 CREATE**

2002 **8.2.1 Overview**

2003 The CREATE operation is used to request the creation of new Resources on the Server. The  
 2004 CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 5.



2005 **Figure 5 – CREATE operation**

2007 **8.2.2 CREATE request**

2008 The CREATE request message is transmitted by the Client to the Server to create a new Resource  
2009 by the Server. The CREATE request message will carry the following parameters:

- 2010 – *fr*: Unique identifier of the Client
- 2011 – *to*: URI of the target Resource responsible for creation of the new Resource.
- 2012 – *ri*: Identifier of the CREATE request.
- 2013 – *cn*: Information of the Resource to be created by the Server.
- 2014 – *cn* will include the URI and Resource Type Property of the Resource to be created.
- 2015 – *cn* may include additional Properties of the Resource to be created.
- 2016 – *op*: CREATE

2017 **8.2.3 Processing by the Server**

2018 Following the receipt of a CREATE request, the Server may validate if the Client has the  
2019 appropriate rights for creating the requested Resource. If the validation is successful, the Server  
2020 creates the requested Resource. The Server caches the value of *ri* parameter in the CREATE  
2021 request for inclusion in the CREATE response message.

2022 **8.2.4 CREATE response**

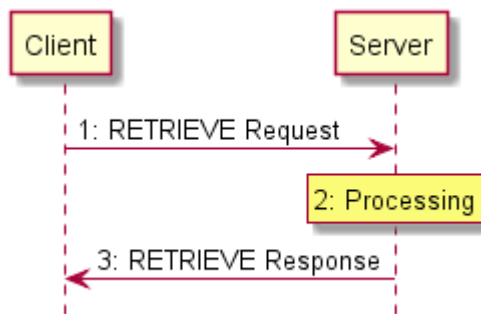
2023 The Server shall transmit a CREATE response message in response to a CREATE request  
2024 message from a Client. The CREATE response message will include the following parameters.

- 2025 – *fr*: Unique identifier of the Server
- 2026 – *to*: Unique identifier of the Client
- 2027 – *ri*: Identifier included in the CREATE request
- 2028 – *cn*: Information of the Resource as created by the Server.
- 2029 – *cn* will include the URI of the created Resource.
- 2030 – *cn* will include the Resource representation of the created Resource.
- 2031 – *rs*: The result of the CREATE operation.

2032 **8.3 RETRIEVE**

2033 **8.3.1 Overview**

2034 The RETRIEVE operation is used to request the current state or representation of a Resource. The  
2035 RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in Figure 6.



2036

2037

**Figure 6 – RETRIEVE operation**

2038 **8.3.2 RETRIEVE request**

2039 RETRIEVE request message is transmitted by the Client to the Server to request the representation  
2040 of a Resource from a Server. The RETRIEVE request message will carry the following parameters.

- 2041 – *fr*: Unique identifier of the Client.
- 2042 – *to*: URI of the Resource the Client is targeting.
- 2043 – *ri*: Identifier of the RETRIEVE request.
- 2044 – *op*: RETRIEVE.

2045 **8.3.3 Processing by the Server**

2046 Following the receipt of a RETRIEVE request, the Server may validate if the Client has the  
2047 appropriate rights for retrieving the requested data and the Properties are readable. The Server  
2048 caches the value of *ri* parameter in the RETRIEVE request for use in the response

2049 **8.3.4 RETRIEVE response**

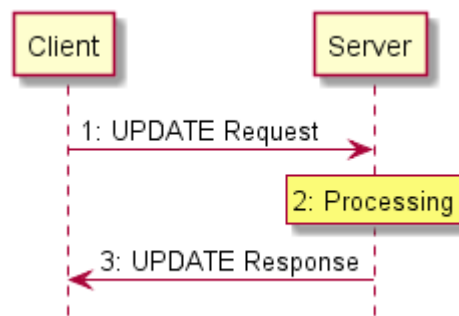
2050 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request  
2051 message from a Client. The RETRIEVE response message will include the following parameters.

- 2052 – *fr*: Unique identifier of the Server.
- 2053 – *to*: Unique identifier of the Client.
- 2054 – *ri*: Identifier included in the RETRIEVE request.
- 2055 – *cn*: Information of the Resource as requested by the Client.
  - 2056 – *cn* should include the URI of the Resource targeted in the RETRIEVE request.
- 2057 – *rs*: The result of the RETRIEVE operation.

2058 **8.4 UPDATE**

2059 **8.4.1 Overview**

2060 The UPDATE operation is either a Partial UPDATE or a complete replacement of the information  
2061 in a Resource in conjunction with the OCF Interface that is also applied to the operation. The  
2062 UPDATE operation is initiated by the Client and consists of three steps, as depicted in Figure 7.



2063  
2064 **Figure 7 – UPDATE operation**

2065 **8.4.2 UPDATE request**

2066 The UPDATE request message is transmitted by the Client to the Server to request the update of  
2067 information of a Resource on the Server. The UPDATE request message will carry the following  
2068 parameters.

- 2069 – *fr*: Unique identifier of the Client.
- 2070 – *to*: URI of the Resource targeted for the information update.

- 2071 – *ri*: Identifier of the UPDATE request.  
2072 – *op*: UPDATE.  
2073 – *cn*: Information, including Properties, of the Resource to be updated at the target Resource.

### 2074 **8.4.3 Processing by the Server**

#### 2075 **8.4.3.1 Overview**

2076 Following the receipt of an UPDATE request, the Server may validate if the Client has the  
2077 appropriate rights for updating the requested data. If the validation is successful the Server updates  
2078 the target Resource information according to the information carried in *cn* parameter of the  
2079 UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE request  
2080 for use in the response.

2081 An UPDATE request that includes Properties that are read-only shall be rejected by the Server with  
2082 an *rs* indicating a bad request.

2083 An UPDATE request shall be applied only to the Properties in the target Resource visible via the  
2084 applied OCF Interface that support the operation. An UPDATE of non-existent Properties is ignored.

2085 An UPDATE request shall be applied to the Properties in the target Resource even if those Property  
2086 Values are the same as the values currently exposed by the target Resource.

#### 2087 **8.4.3.2 Resource monitoring by the Server**

2088 The Server shall monitor the state the Resource identified in the Observe request from the Client.  
2089 Anytime there is a change in the state of the Observed Resource or an UPDATE operation applied  
2090 to the Resource, the Server sends another RETRIEVE response with the Observe indication. The  
2091 mechanism does not allow the Client to specify any bounds or limits which trigger a notification,  
2092 the decision is left entirely to the Server.

#### 2093 **8.4.3.3 Additional RETRIEVE responses with Observe indication**

2094 The Server shall transmit updated RETRIEVE response messages following Observed changes in  
2095 the state of the Resources requested by the Client. The RETRIEVE response message shall include  
2096 the parameters listed in 11.3.2.4.

#### 2097 **8.4.4 UPDATE response**

2098 The UPDATE response message will include the following parameters:

- 2099 – *fr*: Unique identifier of the Server.  
2100 – *to*: Unique identifier of the Client.  
2101 – *ri*: Identifier included in the UPDATE request.  
2102 – *rs*: The result of the UPDATE request.

2103 The UPDATE response message may also include the following parameters:

- 2104 – *cn*: The Resource representation following processing of the UPDATE request.

### 2105 **8.5 DELETE**

#### 2106 **8.5.1 Overview**

2107 The DELETE operation is used to request the removal of a Resource. The DELETE operation is  
2108 initiated by the Client and consists of three steps, as depicted in Figure 8.



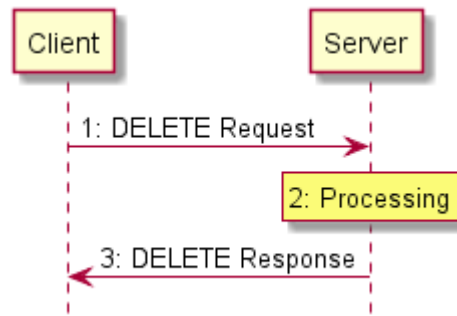


Figure 8 – DELETE operation

2109

2110

2111 **8.5.2 DELETE request**

2112 DELETE request message is transmitted by the Client to the Server to delete a Resource on the  
 2113 Server. The DELETE request message will carry the following parameters:

- 2114 – *fr*: Unique identifier of the Client.
- 2115 – *to*: URI of the target Resource which is the target of deletion.
- 2116 – *ri*: Identifier of the DELETE request.
- 2117 – *op*: DELETE.

2118 **8.5.3 Processing by the Server**

2119 Following the receipt of a DELETE request, the Server may validate if the Client has the appropriate  
 2120 rights for deleting the identified Resource, and whether the identified Resource exists. If the  
 2121 validation is successful, the Server removes the requested Resource and deletes all the associated  
 2122 information. The Server caches the value of *ri* parameter in the DELETE request for use in the  
 2123 response.

2124 **8.5.4 DELETE response**

2125 The Server shall transmit a DELETE response message in response to a DELETE request message  
 2126 from a Client. The DELETE response message will include the following parameters.

- 2127 – *fr*: Unique identifier of the Server.
- 2128 – *to*: Unique identifier of the Client.
- 2129 – *ri*: Identifier included in the DELETE request.
- 2130 – *rs*: The result of the DELETE operation.

2131 **8.6 NOTIFY**

2132 **8.6.1 Overview**

2133 The NOTIFY operation is used to request asynchronous notification of state changes. Complete  
 2134 description of the NOTIFY operation is provided in 11.3. The NOTIFY operation uses the  
 2135 NOTIFICATION response message which is defined here.

2136 **8.6.2 NOTIFICATION response**

2137 The NOTIFICATION response message is sent by a Server to notify the URLs identified by the  
 2138 Client of a state change. The NOTIFICATION response message carries the following parameters.

- 2139 – *fr*: Unique identifier of the Server.
- 2140 – *to*: URI of the Resource target of the NOTIFICATION message.
- 2141 – *ri*: Identifier included in the CREATE request.

- 2142 – *op*: NOTIFY.
- 2143 – *cn*: The updated state of the Resource.

## 2144 **9 Network and connectivity**

### 2145 **9.1 Introduction**

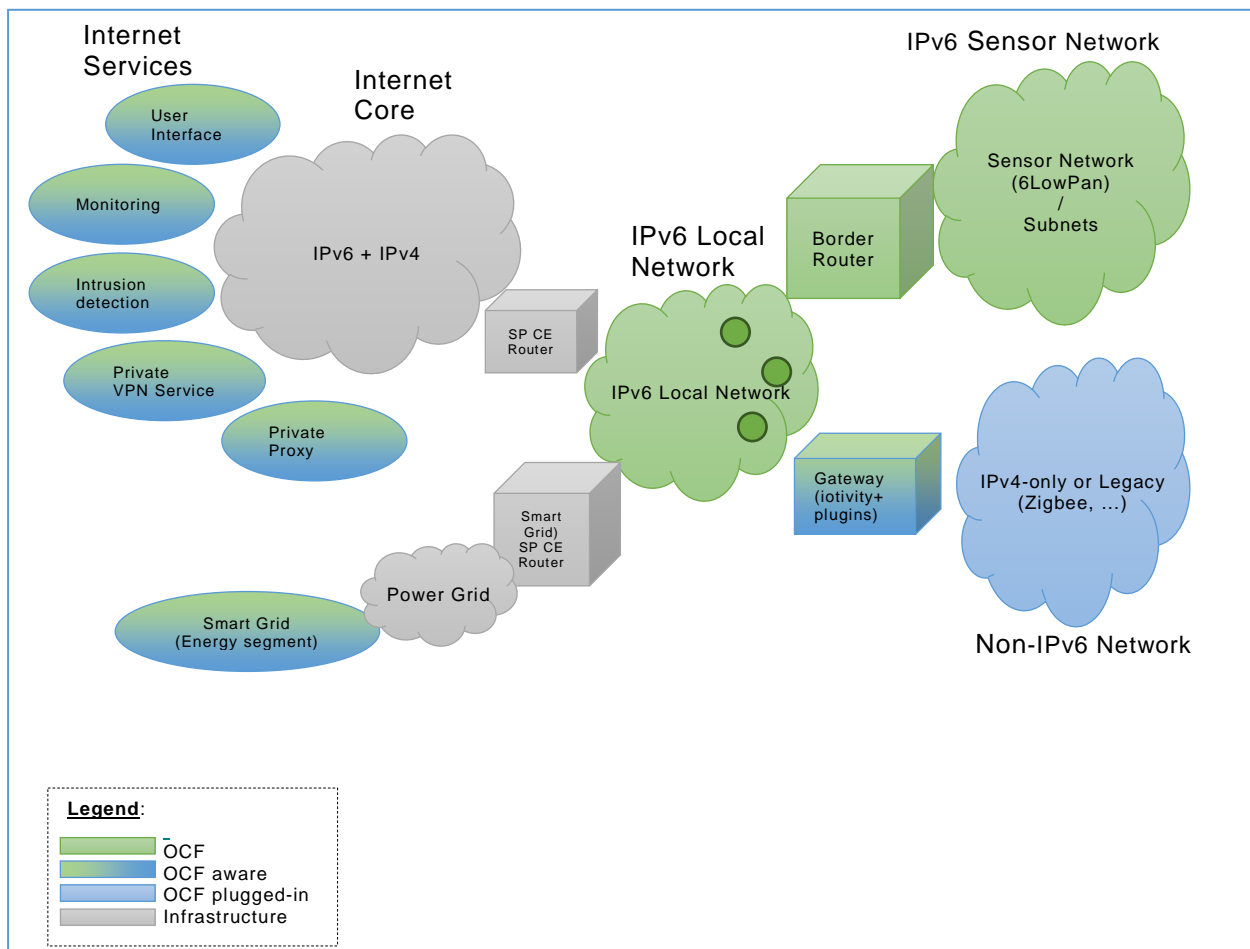
2146 The Internet of Things is comprised of a wide range of applications which sense and actuate the  
2147 physical world with a broad spectrum of device and network capabilities: from battery powered  
2148 nodes transmitting 100 bytes per day and able to last 10 years on a coin cell battery, to mains  
2149 powered nodes able to maintain Megabit video streams. It is estimated that many 10s of billions of  
2150 IoT devices will be deployed over the coming years.

2151 It is desirable that the connectivity options be adapted to the IP layer. To that end, IETF has  
2152 completed considerable work to adapt Bluetooth®, Wi-Fi, 802.15.4, LPWAN, etc. to IPv6. These  
2153 adaptations, plus the larger address space and improved address management capabilities, make  
2154 IPv6 the clear choice for the OCF network layer technology.

### 2155 **9.2 Architecture**

2156 While the aging IPv4 centric network has evolved to support complex topologies, its deployment  
2157 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More  
2158 complex network topologies, often seen in residential home, are mostly introduced through the  
2159 acquisition of additional home network devices, which rely on technologies like private Network  
2160 Address Translation (NAT). These technologies require expert assistance to set up correctly and  
2161 should be avoided in a home network as they most often result in breakage of constructs like  
2162 routing, naming and discovery services.

2163 The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices  
2164 and associated routers, but also new services introducing additional edge routers. All these new  
2165 requirements require advance architectural constructs to address complex network topologies like  
2166 the one shown in Figure 9.



2167

2168

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

2169 In terms of IETF RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further  
2170 implement various specializations of those roles:

- 2171 – A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.
- 2172 – Nodes limited in processing power, memory, non-volatile storage or transmission capacity  
2173 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).  
2174 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU  
2175 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, and Near Field Communication (NFC).
- 2176 – A node may translate and route messaging between IPv6 and non-IPv6 networks.

2177 **9.3 IPv6 network layer requirements**

2178 **9.3.1 Introduction**

2179 Projections indicate that many 10s of billions of new IoT endpoints and related services will be  
2180 brought online in the next few years. These endpoint's capabilities will span from battery powered  
2181 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating  
2182 over Ethernet and WiFi links.

2183 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide  
2184 variety of applications such as Web browsing, email, voice, video, and critical system monitoring  
2185 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which  
2186 is that available address space has been consumed.

2187 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF  
2188 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 2189 – Larger address space. Side-effect: greatly reduce the need for NATs.
- 2190 – More flexible addressing architecture. Multiple addresses and types per interface: Link-local,  
2191 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed  
2192 networks, better re-numbering capability, etc.
- 2193 – More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 2194 – Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 2195 – All major consumer operating systems (IoT, Android, Windows, Linux) are already IPv6 enabled.
- 2196 – Major Service Providers around the globe are deploying IPv6.

### 2197 **9.3.2 IPv6 node requirements**

#### 2198 **9.3.2.1 Introduction**

2199 In order to ensure network layer services interoperability from node to node, mandating a common  
2200 network layer across all nodes is vital. The protocol should enable the network to be: secure,  
2201 manageable, and scalable and to include constrained and self-organizing meshed nodes. OCF  
2202 mandates IPv6 as the common network layer protocol to ensure interoperability across all Devices.  
2203 More capable Devices may also include additional protocols creating multiple-stack Devices. The  
2204 remainder of this clause will focus on interoperability requirements for IPv6 hosts, IPv6 constrained  
2205 hosts and IPv6 routers. The various protocol translation permutations included in multi-stack  
2206 gateway devices may be addresses in subsequent addendums of this document.

#### 2207 **9.3.2.2 IP Layer**

2208 An IPv6 node shall support IPv6 and it shall conform to the requirements as specified in  
2209 IETF RFC 6434.

## 2210 **10 OCF Endpoint**

### 2211 **10.1 OCF Endpoint definition**

2212 The specific definition of an OCF Endpoint depends on the Transport Protocol Suite being used.  
2213 For the example of CoAP over UDP over IPv6, the OCF Endpoint is identified by an IPv6 address  
2214 and UDP port number.

2215 Each Device shall associate with at least one OCF Endpoint with which it can exchange request  
2216 and response messages. When a message is sent to an OCF Endpoint, it shall be delivered to the  
2217 Device which is associated with the OCF Endpoint. When a request message is delivered to an  
2218 OCF Endpoint, path component is enough to locate the target Resource.

2219 A Device can be associated with multiple OCF Endpoints. For example, n Device can have several  
2220 IP addresses or port numbers or support both CoAP and HTTP transfer protocol. Different  
2221 Resources in n Device may be accessed with the same OCF Endpoint or need different ones. Some  
2222 Resources may use one OCF Endpoint and others a different one. It depends on an implementation.

2223 On the other hand, an OCF Endpoint can be shared among multiple Devices, only when there is a  
2224 way to clearly designate the target Resource with request URI. For example, when multiple CoAP  
2225 servers use uniquely different URI paths for all their hosted Resources, and the CoAP  
2226 implementation demultiplexes by path, they can share the same CoAP OCF Endpoint. However,  
2227 this is not possible in this version of the document, because a pre-determined URI (e.g. "/oic/d") is  
2228 mandatory for some mandatory Resources (e.g. "oic.wk.d").

2229 **10.2 OCF Endpoint information**

2230 **10.2.1 Introduction**

2231 OCF Endpoint is represented by OCF Endpoint information which consists of two items of key-  
2232 value pair, "ep" and "pri".

2233 **10.2.2 "ep"**

2234 "ep" represents Transport Protocol Suite and OCF Endpoint Locator specified as follows:

2235 – *Transport Protocol Suite* - a combination of protocols (e.g. CoAP + UDP + IPv6) with which  
2236 request and response messages can be exchanged for RESTful transaction (i.e. CRUDN). A  
2237 Transport Protocol Suite shall be indicated by a URI scheme name. All scheme names  
2238 supported by this document are IANA registered, these are listed in Table 18. A vendor may  
2239 also make use of a non-IANA registered scheme name for their own use (e.g.  
2240 "com.example.foo"), this shall follow the syntax for such scheme names defined by  
2241 IETF RFC 7595. The behaviour of a vendor-defined scheme name is undefined by this  
2242 document. All OCF defined Resource Types when exposing OCF Endpoint Information in an  
2243 "eps" (see 10.2.4) shall include at least one "ep" with a Transport Protocol Suite as defined in  
2244 Table 18.

2245 – *OCF Endpoint Locator* – an address (e.g. IPv6 address + Port number) or an indirect identifier  
2246 (e.g., DNS name) resolvable to an IP address, through which a message can be sent to the  
2247 OCF Endpoint and in turn associated Device. The OCF Endpoint Locator for "coap" and "coaps"  
2248 shall be specified as "IP address: port number". The OCF Endpoint Locator for "coap+tcp" or  
2249 "coaps+tcp" shall be specified as "IP address: port number" or "DNS name: port number" or  
2250 "DNS name" such that the DNS name shall be resolved to a valid IP address for the target  
2251 Resource with a name resolution service (i.e., DNS). For the 3rd case, when the port number  
2252 is omitted, the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and for  
2253 "coaps+tcp") scheme respectively as defined in IETF RFC 8323. Temporary addresses should  
2254 not be used because OCF Endpoint Locators are for the purpose of accepting incoming  
2255 sessions, whereas temporary addresses are for initiating outgoing sessions (IETF RFC 4941).  
2256 Moreover, its inclusion in "/oic/res" can cause a privacy concern (IETF RFC 7721).

2257 "ep" shall have as its value a URI (as specified in IETF RFC 3986) with the scheme component  
2258 indicating Transport Protocol Suite and the authority component indicating the OCF Endpoint  
2259 Locator.

2260 An "ep" example for "coap" and "coaps" is as illustrated:

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

2261 An "ep" example for "coap+tcp" and "coaps+tcp" is as illustrated:

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

2262 The current list of "ep" with corresponding Transport Protocol Suite is shown in Table 18:

**Table 18 – "ep" value for Transport Protocol Suite**

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

2264

**2265 10.2.3 "pri"**

2266 When there are multiple OCF Endpoints, "pri" indicates the priority among them.

2267 "pri" shall be represented as a positive integer (e.g. "pri": 1) and the lower the value, the higher the  
2268 priority.

2269 The default "pri" value is 1, i.e. when "pri" is not present, it shall be equivalent to "pri": 1.

**2270 10.2.4 OCF Endpoint information in "eps" Parameter**

2271 To carry OCF Endpoint information, a new Link Parameter "eps" is defined in 7.8.2.2.5. "eps" has  
2272 an array of items as its value and each item represents OCF Endpoint information with two key-  
2273 value pairs, "ep" and "pri", of which "ep" is mandatory and "pri" is optional.

2274 OCF Endpoint Information in an "eps" Parameter is valid for the target Resource of the Link, i.e.,  
2275 the Resource referred by "href" Parameter. OCF Endpoint information in an "eps" Parameter may  
2276 be used to access other Resources on the Device, but such access is not guaranteed.

2277 A Client may resolve the "ep" value to an IP address for the target Resource, i.e., the address to  
2278 access the Device which hosts the target Resource. A valid (transfer protocol) URI for the target  
2279 Resource can be constructed with the scheme, host and port components from the "ep" value and  
2280 the "path" component from the "href" value.

2281 Links with an "eps":

```

2282 {
2283   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9 ",
2284   "href": "/myLightSwitch",
2285   "rt": ["oic.r.switch.binary"],
2286   "if": ["oic.if.a", "oic.if.baseline"],
2287   "p": {"bm": 3},
2288   "eps": [
2289     {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
2290     {"ep": "coaps://[fe80::b1d6]:1122"}
2291   ]
2292 }
2293

```

```

2294 {
2295   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2296   "href": "/myTemperature",
2297   "rt": ["oic.r.temperature"],
2298   "if": ["oic.if.a", "oic.if.baseline"],
2299   "p": {"bm": 3},
2300   "eps": [
2301     {"ep": "coap+tcp://foo.bar.com", "pri": 2},
2302     {"ep": "coaps+tcp://foo.bar.com:1122"}
2303   ]
2304 }

```

2305 In the previous example, "anchor" represents the hosting Device, "href", target Resource and "eps"  
 2306 the two OCF Endpoints for the target Resource. The (fully-qualified) URIs for the target Resource  
 2307 are as illustrated:

```

2308 coap://[fe80::b1d6]:1111/myLightSwitch
2309 coaps://[fe80::b1d6]:1122/myLightSwitch
2310 coap+tcp://foo.bar.com:5683/myTemperature

```

2311 coaps+tcp://foo.bar.com:1122/myTemperature If the target Resource of a Link requires a secure  
 2312 connection (e.g. CoAPS), "eps" Parameter shall be used to indicate the necessary information (e.g.  
 2313 port number) in OCF 1.0 payload. For optional backward compatibility with OIC 1.1, the "sec" and  
 2314 "port" shall only be used in OIC 1.1 payload.

### 2315 10.3 OCF Endpoint discovery

#### 2316 10.3.1 Introduction

2317 OCF Endpoint discovery is defined as the process for a Client to acquire the OCF Endpoint  
 2318 information for Device or Resource.

#### 2319 10.3.2 Implicit discovery

2320 If a Device is the source of a CoAP message (e.g. "/oic/res" response), the source IP address and  
 2321 port number may be combined to form the OCF Endpoint Locator for the Device. Along with a  
 2322 "coap" scheme and default "pri" value, OCF Endpoint information for the Device may be constructed.

2323 In other words, a "/oic/res" response message with CoAP may implicitly carry the OCF Endpoint  
 2324 information of the responding Device and in turn all the hosted Resources, which may be accessed  
 2325 with the same transfer protocol of CoAP. In the absence of an "eps" Parameter, a Client shall be  
 2326 able to utilize implicit discovery to access the target Resource.

#### 2327 10.3.3 Explicit discovery with "/oic/res" response

2328 OCF Endpoint information may be explicitly indicated with the "eps" Parameter of the Links in  
 2329 "/oic/res".

2330 As in 10.3.2, an "/oic/res" response may implicitly indicate the OCF Endpoint information for some  
 2331 Resources hosted by the responding Device. However implicit discovery, i.e., inference of OCF  
 2332 Endpoint information from CoAP response message, may not work for some Resources on the  
 2333 same Device. For example, some Resources may allow only secure access via CoAPS which  
 2334 requires the "eps" Parameter to indicate the port number. Moreover "/oic/res" may expose a target  
 2335 Resource which belongs to another Device.

2336 When the OCF Endpoint for a target Resource of a Link cannot be implicitly inferred, the "eps"  
 2337 Parameter shall be included to provide explicit OCF Endpoint information with which a Client can  
 2338 access the target Resource. In the presence of the "eps" Parameter, a Client shall be able to utilize  
 2339 it to access the target Resource. For "coap" and "coaps", a Client may use the IP address in the  
 2340 "ep" value in the "eps" Parameter to access the target Resource. For "coap+tcp" and "coaps+tcp",  
 2341 a Client may use the IP address in the "eps" Parameter or resolve the DNS name in the "eps"  
 2342 Parameter to acquire a valid IP address for the target Resource. If "eps" Parameter omits the port

2343 number, then the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and  
2344 "coaps+tcp") scheme as defined in IETF RFC 8323. To access the target Resource of a Link, a  
2345 Client may use the "eps" Parameter in the Link, if it is present and fall back on implicit discovery if  
2346 not.

2347 This is an example of an "/oic/res" response from a Device having the "eps" Parameter in Links.

```
2348  
2349 [  
2350   {  
2351     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",  
2352     "href": "/oic/res",  
2353     "rel": "self",  
2354     "rt": ["oic.wk.res"],  
2355     "if": ["oic.if.ll", "oic.if.baseline"],  
2356     "p": {"bm": 3},  
2357     "eps": [  
2358       {"ep": "coap://[2001:db8:a::b1d4]:55555"},  
2359       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}  
2360     ]  
2361   },  
2362   {  
2363     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",  
2364     "href": "/oic/d",  
2365     "rt": ["oic.wk.d"],  
2366     "if": ["oic.if.r", "oic.if.baseline"],  
2367     "p": {"bm": 3},  
2368     "eps": [  
2369       {"ep": "coap://[2001:db8:a::b1d4]:55555"},  
2370       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}  
2371     ]  
2372   },  
2373   {  
2374     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",  
2375     "href": "/oic/p",  
2376     "rt": ["oic.wk.p"],  
2377     "if": ["oic.if.r", "oic.if.baseline"],  
2378     "p": {"bm": 3},  
2379     "eps": [  
2380       {"ep": "coap://[2001:db8:a::b1d4]:55555"},  
2381       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}  
2382     ]  
2383   },  
2384   {  
2385     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",  
2386     "href": "/oic/sec/doxm",  
2387     "rt": ["oic.r.doxm"],  
2388     "if": ["oic.if.baseline"],  
2389     "p": {"bm": 1},  
2390     "eps": [  
2391       {"ep": "coap://[2001:db8:a::b1d4]:55555"},  
2392       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}  
2393     ]  
2394   },  
2395   {  
2396     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",  
2397     "href": "/oic/sec/pstat",  
2398     "rt": ["oic.r.pstat"],  
2399     "if": ["oic.if.baseline"],  
2400     "p": {"bm": 1},  
2401     "eps": [  
2402       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}  
2403     ]  
2404   }  
2405 ]
```



```

2403     ]
2404   },
2405   {
2406     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2407     "href": "/oic/sec/cred",
2408     "rt": ["oic.r.cred"],
2409     "if": ["oic.if.baseline"],
2410     "p": {"bm": 1},
2411     "eps": [
2412       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2413     ]
2414   },
2415   {
2416     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2417     "href": "/oic/sec/acl2",
2418     "rt": ["oic.r.acl2"],
2419     "if": ["oic.if.baseline"],
2420     "p": {"bm": 1},
2421     "eps": [
2422       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2423     ]
2424   },
2425   {
2426     "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2427     "href": "/myIntrospection",
2428     "rt": ["oic.wk.introspection"],
2429     "if": ["oic.if.r", "oic.if.baseline"],
2430     "p": {"bm": 3},
2431     "eps": [
2432       {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2433     ]
2434   },
2435   {
2436     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2437     "href": "/myLight",
2438     "rt": ["oic.r.switch.binary"],
2439     "if": ["oic.if.a", "oic.if.baseline"],
2440     "p": {"bm": 3},
2441     "eps": [
2442       {"ep": "coaps://[2001:db8:a::b1d4]:22222"}
2443     ]
2444   }
2445 ]
2446

```

2447 The exact format of the "/oic/res" response and a way for a Client to acquire a "/oic/res" response  
2448 message is specified in Annex A and 11.2.4 respectively.

## 2449 11 Functional interactions

### 2450 11.1 Introduction

2451 The functional interactions between a Client and a Server are described in 11.1 through 11.4  
2452 respectively. The functional interactions use CRUDN messages (clause 8) and include Discovery,  
2453 Notification, and Device management. These functions require support of core defined Resources  
2454 as defined in Table 19.

2455 **Table 19 – 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
<b>Implementation defined</b>	Introspection	"oic.wk.introspection"	Introspection	Yes

2456

2457 **11.2 Resource discovery**

2458 **11.2.1 Introduction**

2459 Discovery is a function which enables OCF Endpoint discovery as well as Resource based  
 2460 discovery. OCF Endpoint discovery is described in detail in clause 10. This clause mainly describes  
 2461 the Resource based discovery.

2462 **11.2.2 Resource based discovery: mechanisms**

2463 **11.2.2.1 Overview**

2464 As part of discovery, a Client may find appropriate information about other OCF peers. This  
 2465 information could be instances of Resources, Resource Types or any other information represented  
 2466 in the Resource model that an OCF peer would want another OCF peer to discover.

2467 At the minimum, Resource based discovery uses the following:

- 2468 – A Resource to enable discovery shall be defined. The representation of that Resource shall  
 2469 contain the information that can be discovered.
- 2470 – The Resource to enable discovery shall be specified and commonly known a-priori. A Device  
 2471 for hosting the Resource to enable discovery shall be identified.
- 2472 – A mechanism and process to publish the information that needs to be discovered with the  
 2473 Resource to enable discovery.
- 2474 – A mechanism and process to access and obtain the information from the Resource to enable  
 2475 discovery. A query may be used in the request to limit the returned information.
- 2476 – A scope for the publication.
- 2477 – A scope for the access.
- 2478 – A policy for visibility of the information.

2479 Depending on the choice of the base aspects, the Framework defines three Resource based  
 2480 discovery mechanisms:

- 2481 – Direct discovery, where the Resources are published locally at the Device hosting the  
 2482 Resources and are discovered through peer inquiry.
- 2483 – Indirect discovery, where Resources are published at a third party assisting with the discovery  
 2484 and peers publish and perform discovery against the Resource to enable discovery on the  
 2485 assisting 3<sup>rd</sup> party.
- 2486 – Advertisement discovery, where the Resource to enable discovery is hosted local to the initiator  
 2487 of the discovery inquiry but remote to the Devices that are publishing discovery information.

2488 A Device shall support direct discovery.

2489 **11.2.2.2 Direct discovery**

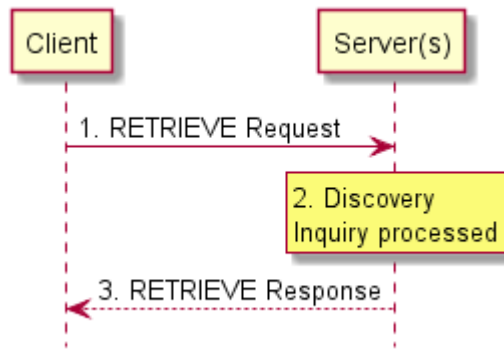
2490 In direct discovery,

- 2491 – The Device that is providing the information shall host the Resource to enable discovery.

- 2492 – The Device publishes the information available for discovery with the local Resource to enable  
2493 discovery (i.e. local scope).
- 2494 – Clients interested in discovering information about this Device shall issue RETRIEVE requests  
2495 directly to the Resource. The request may be made as a unicast or multicast. The request may  
2496 be generic or may be qualified or limited by using appropriate queries in the request.
- 2497 – The Server Device that receives the request shall send a response with the discovered  
2498 information directly back to the requesting Client Device.
- 2499 – The information that is included in the request is determined by the policies set for the Resource  
2500 to be discovered locally on the responding Device.

2501 **11.2.3 Resource based discovery: Finding information**

2502 The discovery process (Figure 10) is initiated as a RETRIEVE request to the Resource to enable  
2503 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as  
2504 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the  
2505 support in the data connectivity layer. The response to the request has the information to be  
2506 discovered based on the policies for that information. The policies can determine which information  
2507 is shared, when and to which requesting agent. The information that can be discovered can be  
2508 Resources, types, configuration and many other standards or custom aspects depending on the  
2509 request to appropriate Resource and the form of request. Optionally the requester may narrow the  
2510 information to be returned in the request using query parameters in the URI query.



2511

2512 **Figure 10 – Resource based discovery: Finding information**

2513

2514 *Discovery Resources*

2515 The following Core Resources shall be implemented on all Devices to support discovery:

- 2516 – "/oic/res" for discovery of Resources.
- 2517 – "/oic/p" for discovery of Platform.
- 2518 – "/oic/d" for discovery of Device information.

2519 Devices shall expose each of "/oic/res", "/oic/d", and "/oic/p" via an unsecured OCF Endpoint.  
2520 Further details for these mandatory Core Resources are described in Table 20.

2521 *Platform Resource*

2522 The OCF recognizes that more than one instance of Device may be hosted on a single Platform.  
2523 Clients need a way to discover and access the information on the Platform. The Core Resource,  
2524 "/oic/p" exposes Platform specific Properties. All instances of Device on the same Platform shall  
2525 have the same values of any Properties exposed (i.e. a Device may choose to expose optional

2526 Properties within "/oic/p" but when exposed the value of that Property should be the same as the  
 2527 value of that Property on all other Devices on that Platform).

2528 *Device Resource*

2529 The Device Resource shall have the pre-defined URI "/oic/d", the Device Resource shall expose  
 2530 the Properties pertaining to a Device as defined in Table 23. The Device Resource shall have a  
 2531 default Resource Type that helps in bootstrapping the interactions with the Device (the default type  
 2532 is described in Table 20).The Device Resource may have one or more Resource Type(s) that are  
 2533 specific to the Device in addition to the default Resource Type or if present overriding the default  
 2534 Resource Type. The base Resource Type "oic.wk.d" defines the Properties that shall be exposed  
 2535 by all Devices. The Device specific Resource Type(s) exposed are dependent on the class of  
 2536 Device (e.g. air conditioner, smoke alarm, etc. Since all the Resource Types of "/oic/d" are not  
 2537 known a priori, the Resource Type(s) of "/oic/d" are determined by discovery through the Core  
 2538 Resource "/oic/res".

2539 **Table 20 – Mandatory discovery Core Resources**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
"/oic/res"	Default	"oic.wk.res"	"oic.if.ll"	The Resource through which the corresponding Server is discovered and introspected for available Resources. "/oic/res" shall expose the Resources that are discoverable on a Device. When a Server receives a RETRIEVE request targeting "/oic/res" (e.g., "GET /oic/res"), it shall respond with the links list of all the Discoverable Resources of itself. The "/oic/d" and "/oic/p" are Discoverable Resources, hence their links are included in "/oic/res" response. The Properties exposed by "/oic/res" are listed in Table 21.	Discovery
"/oic/p"	Platform	"oic.wk.p"	"oic.if.r"	The Discoverable Resource through which Platform specific information is discovered. The Properties exposed by "/oic/p" are listed in Table 24	Discovery
"/oic/d"	Device	"oic.wk.d" and/or one or more Device Specific Resource Type ID(s)	"oic.if.r"	The discoverable via "/oic/res" Resource which exposes Properties specific to the Device instance. The Properties exposed by "/oic/d" are listed in Table 23.	Discovery

2540 Table 21 defines "oic.wk.res" Resource Type.

2541 **Table 21 – "oic.wk.res" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
Name	"n"	string	N/A	N/A	R	No	Human-friendly name defined by the vendor
Links	"links"	array	See 7.8.2	N/A	R	Yes	The array of Links describes the URI, supported Resource Types and OCF

							Interfaces, and access policy.
--	--	--	--	--	--	--	--------------------------------

- 2542
- 2543 A Device shall support CoAP based discovery as the baseline discovery mechanism (see 11.2.5).
- 2544 The "/oic/res" shall list all Resources that are indicated as discoverable (see 11.2). Also the  
2545 following architecture Resource Types shall be listed:
- 2546 – Introspection Resource indicated with an "rt" value of "oic.wk.introspection".
  - 2547 – "/oic/p" indicated with an "rt" value of "oic.wk.p".
  - 2548 – "/oic/d" indicated with an "rt" value of "oic.wk.d"
  - 2549 – "/oic/sec/doxm" indicated with an "rt" value of "oic.r.doxm" as defined in ISO/IEC 30118-2:2018.
  - 2550 – "/oic/sec/pstat" indicated with an "rt" value of "oic.r.pstat" as defined in ISO/IEC 30118-2:2018.
  - 2551 – "/oic/sec/acl2" indicated with an "rt" value of "oic.r.acl2" as defined in ISO/IEC 30118-2:2018.
  - 2552 – "/oic/sec/cred" indicated with an "rt" value of "oic.r.cred" as defined in ISO/IEC 30118-2:2018.
- 2553 Conditionally required:
- 2554 – "/oic/res" with an "rt" value of "oic.wk.res" as self-reference, on the condition that "oic/res" has  
2555 to signal that it is Observable by a Client.
- 2556 The Introspection Resource is only applicable for Devices that host Vertical Resource Types (e.g.  
2557 "oic.r.switch.binary") or vendor-defined Resource Types. Devices that only host Resources  
2558 required to onboard the Device as a Client do not have to implement the Introspection Resource.
- 2559 Table 22 provides an OCF registry for protocol schemes.

2560 **Table 22 – Protocol scheme registry**

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

- 2561
- 2562 NOTE The discovery of an OCF Endpoint used by a specific protocol is out of scope. The mechanism used by a Client  
2563 to form requests in a different messaging protocol other than discovery is out of scope.
- 2564 The following applies to the use of "/oic/d":
- 2565 – A vertical may choose to extend the list of Properties defined by the Resource Type "oic.wk.d".  
2566 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The  
2567 mandatory Properties defined in Table 23 shall always be present.
  - 2568 – A Device may choose to expose a separate, Discoverable Resource with its Resource Type ID  
2569 set to a Device Type. In this case the Resource is equivalent to an instance of "oic.wk.d" and  
2570 adheres to the definition thereof. As such the Resource shall at a minimum expose the  
2571 mandatory Properties of "oic.wk.d". In the case where the Resource tagged in this manner is  
2572 defined to be an instance of a Collection in accordance with 7.8.3 then the Resources that are

2573 part of that Collection shall at a minimum include the Resource Types mandated for the Device  
 2574 Type.

2575 Table 23 "oic.wk.d" Resource Type definition defines the base Resource Type for the "/oic/d"  
 2576 Resource.

2577 **Table 23 – "oic.wk.d" Resource Type definition**

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

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

							Property, otherwise this Property shall not be included.
--	--	--	--	--	--	--	--

2578 Table 24 defines "oic.wk.p" Resource Type.

2579

**Table 24 – "oic.wk.p" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
<b>Platform ID</b>	"pi"	"uuid"	N/A	N/A	R	Yes	Unique identifier for the physical Platform (UUID); this shall be a UUID in accordance with IETF RFC 4122. It is recommended that the UUID be created using the random generation scheme (version 4 UUID) specific in the RFC. Handling privacy-sensitivity for the "pi" Property, refer to clause 13.16 in ISO/IEC 30118-2:2018.
<b>Manufacturer Name</b>	"mnmn"	"string"	N/A	N/A	R	Yes	Name of manufacturer.
<b>Manufacturer Details Link</b>	"mnml"	"uri"	N/A	N/A	R	No	Reference to manufacturer, represented as a URI.
<b>Model Number</b>	"mnmo"	"string"	N/A	N/A	R	No	Model number as designated by manufacturer.
<b>Date of Manufacture</b>	"mndt"	"date"	N/A	Time	R	No	Manufacturing date of Platform.
<b>Serial number</b>	"mnsel"	"string"	N/A	s	R	No	Serial number of the Platform, may be unique for each Platform of the same model number.
<b>Platform Version</b>	"mnpv"	"string"	N/A	N/A	R	No	Version of Platform – string (defined by manufacturer).
<b>OS Version</b>	"mnos"	"string"	N/A	N/A	R	No	Version of Platform resident OS – string (defined by manufacturer).
<b>Hardware Version</b>	"mnhw"	"string"	N/A	N/A	R	No	Version of Platform hardware.
<b>Firmware version</b>	"mnfv"	"string"	N/A	N/A	R	No	Version of Platform firmware.
<b>Support link</b>	"mnsi"	"uri"	N/A	N/A	R	No	URI that points to support information from manufacturer.
<b>SystemTime</b>	"st"	"date-time"	N/A	N/A	R	No	Reference time for the Platform.
<b>Vendor ID</b>	"vid"	"string"	N/A	N/A	R	No	Vendor defined string for the Platform. The string is freeform and up



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

2580 **11.2.4 Resource discovery using "/oic/res"**

2581 Discovery using "/oic/res" is the default discovery mechanism that shall be supported by all Devices  
2582 as follows:

- 2583 – Every Device updates its local "/oic/res" with the Resources that are discoverable (see 7.3.2.2).  
2584 Every time a new Resource is instantiated on the Device and if that Resource is discoverable  
2585 by a remote Device then that Resource is published with the "/oic/res" Resource that is local to  
2586 the Device (as the instantiated Resource).
- 2587 – A Device wanting to discover Resources or Resource Types on one or more remote Devices  
2588 makes a RETRIEVE request to the "/oic/res" on the remote Devices. This request may be sent  
2589 multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request may  
2590 optionally be restricted using appropriate clauses in the query portion of the request. Queries  
2591 may select based on Resource Types, OCF Interfaces, or Properties.
- 2592 – The query applies to the representation of the Resources. "/oic/res" is the only Resource whose  
2593 representation has "rt". So "/oic/res" is the only Resource that can be used for Multicast  
2594 discovery at the transport protocol layer.
- 2595 – The Device receiving the RETRIEVE request responds with a list of Resources, the Resource  
2596 Type of each of the Resources and the OCF Interfaces that each Resource supports.  
2597 Additionally, information on the policies active on the Resource can also be sent. The policy  
2598 supported includes Observability and discoverability.
- 2599 – The receiving Device may do a deeper discovery based on the Resources returned in the  
2600 request to "/oic/res".

2601 The information that is returned on discovery against "/oic/res" is at the minimum:

- 2602 – The URI (relative or fully qualified URL) of the Resource.
- 2603 – The Resource Type(s) of each Resource. More than one Resource Type may be returned if the  
2604 Resource enables more than one type. To access Resources of multiple types, the specific  
2605 Resource Type that is targeted shall be specified in the request.
- 2606 – The OCF Interfaces supported by that Resource. Multiple OCF Interfaces may be returned. To  
2607 access a specific OCF Interface that OCF Interface shall be specified in the request. If the OCF  
2608 Interface is not specified, then the Default OCF Interface is assumed.

2609 For Clients that do include the OCF-Accept-Content-Format-Version option, an "/oic/res" response  
2610 includes an array of Links to conform to IETF RFC 6690. Each Link shall use an "eps" Parameter  
2611 to provide the information for an encrypted connection and carry "anchor" of the value OCF URI  
2612 where the authority component of <deviceID> indicates the Device hosting the target Resource.

2613 The OpenAPI 2.0 file for discovery using "/oic/res" is described in Annex A. Also refer to clause 10  
2614 (OCF Endpoint discovery) for details of Multicast discovery using "/oic/res" on a CoAP transport.

2615 An example Device might return the following to Clients that request with the Content Format of  
2616 "application/vnd.ocf+cbor" in Accept Option:

```

2617 [
2618   {
2619     "href": "/oic/res",
2620     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989/oic/res",
2621     "rel": "self",
2622     "rt": ["oic.wk.res"],
2623     "if": ["oic.if.ll", "oic.if.baseline"],
2624     "p": {"bm": 3},
2625     "eps": [{"ep": "coap://[fe80::b1d6]:44444"}]
2626   },
2627   {
2628     "href": "/oic/p",
2629     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2630     "rt": ["oic.wk.p"],
2631     "if": ["oic.if.r", "oic.if.baseline"],
2632     "p": {"bm": 3},
2633     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2634             {"ep": "coaps://[fe80::b1d6]:11111"}]
2635   },
2636 },
2637 {
2638   "href": "/oic/d",
2639   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2640   "rt": ["oic.wk.d"],
2641   "if": ["oic.if.r", "oic.if.baseline"],
2642   "p": {"bm": 3},
2643   "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2644           {"ep": "coaps://[fe80::b1d6]:11111"}]
2645 },
2646 },
2647 {
2648   "href": "/myLightSwitch",
2649   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2650   "rt": ["oic.r.switch.binary"],
2651   "if": ["oic.if.a", "oic.if.baseline"],
2652   "p": {"bm": 3},
2653   "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2654           {"ep": "coaps://[fe80::b1d6]:11111"}]
2655 },
2656 ]
2657 ]

```

2658 After performing discovery using "/oic/res", Clients may discover additional details about Server by  
2659 performing discovery using "/oic/p", etc. If a Client already knows about Server it may discover  
2660 using other Resources without going through the discovery of "/oic/res".

### 2661 11.2.5 Multicast discovery using "/oic/res"

2662 Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices shall  
2663 support use of CoAP multicast to allow retrieving the "/oic/res" Resource from an unsecured OCF  
2664 Endpoint on the Device. Clients may support use of CoAP multicast to retrieve the "/oic/res"  
2665 Resource from other Devices. The CoAP multicast retrieval of "/oic/res" supports filtering Links  
2666 based on the "rt" Property in the Links:

- 2667 – If the discovery request is intended for a specific Resource Type including as part of a multi-  
2668 value Resource Type, the query parameter "rt" shall be included in the request (see 6.2.2) with  
2669 its value set to the desired Resource Type. Only Devices hosting the Resource Type shall  
2670 respond to the discovery request.
- 2671 – When the "rt" query parameter is omitted, all Devices shall respond to the discovery request.

2672 **11.3 Notification**

2673 **11.3.1 Overview**

2674 A Server shall support NOTIFY operation to enable a Client to request and be notified of desired  
2675 states of one or more Resources in an asynchronous manner. 11.3.2 specifies the Observe  
2676 mechanism in which updates are delivered to the requester.

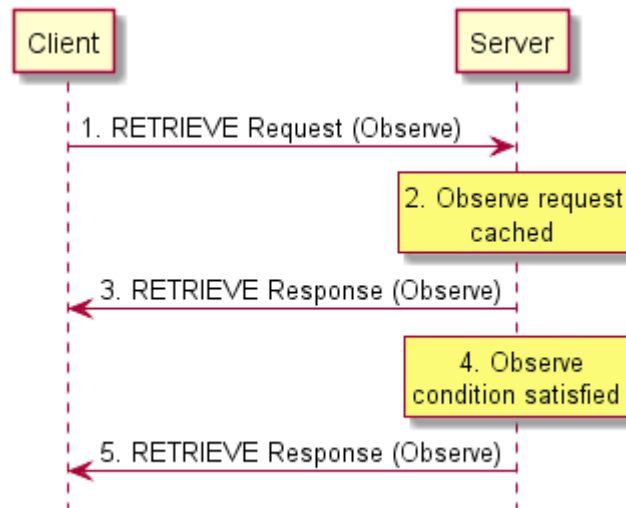
2677 **11.3.2 Observe**

2678 **11.3.2.1 Overview**

2679 In the Observe mechanism the Client utilizes the RETRIEVE operation to require the Server for  
2680 updates in case of Resource state changes. The Observe mechanism consists of five steps which  
2681 are depicted in Figure 11.

2682 NOTE the Observe mechanism can only be used for a resource with a Property of Observable (see 7.3.2.2).

2683



2684

2685

2686 **Figure 11 – Observe Mechanism**

2687 **11.3.2.2 RETRIEVE request with Observe indication**

2688 The Client transmits a RETRIEVE request message to the Server to request updates for the  
2689 Resource on the Server if there is a state change. The RETRIEVE request message carries the  
2690 following parameters:

- 2691 – *fr*: Unique identifier of the Client.
- 2692 – *to*: Resource that the Client is requesting to Observe.
- 2693 – *ri*: Identifier of the RETRIEVE operation.
- 2694 – *op*: RETRIEVE.
- 2695 – *obs*: Indication for Observe operation.

2696 **11.3.2.3 Processing by the Server**

2697 Following the receipt of the RETRIEVE request, the Server may validate if the Client has the  
2698 appropriate rights for the requested operation and the Properties are readable and Observable. If  
2699 the validation is successful, the Server caches the information related to the Observe request. The

2700 Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial  
2701 response and future responses in case of a change of state.

#### 2702 **11.3.2.4 RETRIEVE response with Observe indication**

2703 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request  
2704 message from a Client. The RETRIEVE response message shall include the following parameters.  
2705 If validation succeeded, the response includes an Observe indication. If not, the Observe indication  
2706 is omitted from the response which signals to the requesting Client that registration for notification  
2707 was not allowed.

2708 The RETRIEVE response message shall include the following parameters:

- 2709 – *fr*: Unique identifier of the Server.
- 2710 – *to*: Unique identifier of the Client.
- 2711 – *ri*: Identifier included in the RETRIEVE operation.
- 2712 – *cn*: Information Resource representation as requested by the Client.
- 2713 – *rs*: The result of the RETRIEVE operation.
- 2714 – *obs*: Indication that the response is made to an Observe operation.

#### 2715 **11.3.2.5 Resource monitoring by the Server**

2716 The Server shall monitor the state the Resource identified in the Observe request from the Client.  
2717 Anytime there is a change in the state of the Observed Resource, the Server sends another  
2718 RETRIEVE response with the Observe indication. The mechanism does not allow the client to  
2719 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

#### 2720 **11.3.2.6 Additional RETRIEVE responses with Observe indication**

2721 The Server shall transmit updated RETRIEVE response messages following Observed changes in  
2722 the state of the Resources indicated by the Client. The RETRIEVE response message shall include  
2723 the parameters listed in 11.3.2.4.

#### 2724 **11.3.2.7 Cancelling Observe**

2725 The Client can explicitly cancel Observe by sending a RETRIEVE request without the Observe  
2726 indication field to the same Resource on the Server which it was Observing. For certain protocol  
2727 mappings, the Client may also be able to cancel an Observe by ceasing to respond to the  
2728 RETRIEVE responses.

### 2729 **11.4 Introspection**

#### 2730 **11.4.1 Overview**

2731 Introspection is a mechanism to announce the capabilities of Resources hosted on the Device.

2732 The intended usage of the Introspection Device Data (IDD) is to enable dynamic Clients e.g. Clients  
2733 that can use the IDD) to generate dynamically a UI or dynamically create translations of the hosted  
2734 Resources to another eco-system. Other usages of Introspection is that the information can be  
2735 used to generate Client code. The IDD is designed to augment the existing data already on the  
2736 wire. This means that existing mechanisms need to be used to get a full overview of what is  
2737 implemented in the Device. For example, the IDD does not convey information about Observability,  
2738 since that is already conveyed with the "p" Property on the Links in "/oic/res" (see 7.8.2.2.2).

2739 The IDD is recommended to be conveyed as static data. Meaning that the data does not change  
2740 during the uptime of a Device. However, when the IDD is not static, the Introspection Resource  
2741 shall be Observable and the url Property Value of "oic.wk.introspection" Resource shall change to  
2742 indicate that the IDD is changed.

2743 The IDD describes the Resources that make up the Device. For the complete list of included  
2744 Resources see Table 19. The IDD is described as a OpenAPI 2.0 in JSON format file. The text in  
2745 the following bulleted list contains OpenAPI 2.0 terms, such as paths, methods etc. The OpenAPI  
2746 2.0 file shall contain the description of the Resources:

- 2747 – The IDD will use the HTTP syntax, e.g., define the CRUDN operation as HTTP methods and  
2748 use the HTTP status codes.
- 2749 – The IDD does not have to define all the status codes that indicate an error situation.
- 2750 – The IDD does not have to define a schema when the status code indicates that there is no  
2751 payload (see HTTP status code 204 as an example).
- 2752 – The paths (URLs) of the Resources in the IDD shall be without the OCF Endpoint description,  
2753 e.g. it shall not be a fully-qualified URL but only the relative path from the OCF Endpoint, aka  
2754 the "href". The relative path may include a query parameter (e.g. "?if=oic.if.ll"), in such cases  
2755 the text following (and including) the "?" delimiter shall be removed before equating to the "href"  
2756 that is conveyed by "/oic/res".
- 2757 – The following Resources shall be excluded in the IDD:
  - 2758 – Resource with Resource Type: "oic.wk.res" unless 3rd party defined or optional Properties  
2759 are implemented.
  - 2760 – Resource with Resource Type: "oic.wk.introspection".
  - 2761 – Resources that handle Wi-Fi Easy Setup, see OCF Easy Wi-Fi Setup.
- 2762 – The following Resources shall be included in the IDD when optional or 3<sup>rd</sup> party defined  
2763 Properties are implemented:
  - 2764 – Resources with type: "oic.wk.p" and "oic.wk.d" (e.g. discovery related Resources).
  - 2765 – Security Virtual Resources from ISO/IEC 30118-2:2018.
- 2766 – When the Device does not expose instances of Vertical Resource Types, and does not have  
2767 any 3<sup>rd</sup> party defined Resources (see 7.8.4.4), and does not need to include Resources in the  
2768 IDD due to other clauses in this clause, then the IDD shall be an empty OpenAPI 2.0 file. An  
2769 example of an empty OpenAPI 2.0 file can be found in found in Annex **B.2**.
- 2770 – All other Resources that are individually addressable by a Client (i.e. the "href" can be resolved  
2771 and at least one operation is supported with a success path response) shall be listed in the IDD.
- 2772 – Per Resource the IDD shall include:
  - 2773 – All implemented methods
    - 2774 – For an OCF defined Resource Type, only the methods that are listed in the OpenAPI 2.0  
2775 definition are allowed to exist in the IDD. For an OCF defined Resource Type, methods  
2776 not listed in the OpenAPI 2.0 definition shall not exist in the IDD. The supported methods  
2777 contained in the IDD shall comply with the listed OCF Interfaces. For example, if the  
2778 POST method is listed in the IDD, then an OCF Interface that allows UPDATE will be  
2779 listed in the IDD.
  - 2780 – Per supported method:
    - 2781 – Implemented query parameters per method.
      - 2782 – This includes the supported OCF Interfaces ("if") as enum values.
    - 2783 – Schemas of the payload for the request and response bodies of the method.
    - 2784 – Where the schema provides the representation of a batch request or response ("oic.if.b")  
2785 the schema shall contain the representations for all Resource Types that may be  
2786 included within the batch representation. The representations shall be provided within  
2787 the IDD itself.
    - 2788 – The schema data shall be conveyed by the OpenAPI 2.0 schema.

- 2789 – The OpenAPI 2.0 schema object shall comply with:
    - 2790 – The schemas shall be fully resolved, e.g. no references shall exist outside the
    - 2791 OpenAPI 2.0 file.
    - 2792 – The schemas shall list which OCF Interfaces are supported on the method.
    - 2793 – The schemas shall list if a Property is optional or required.
    - 2794 – The schemas shall include all Property validation keywords. Where an enum is
    - 2795 defined the enum shall contain the values supported by the Device. When vendor
    - 2796 defined extensions exist to the enum (defined in accordance to 7.8.4.4) these shall
    - 2797 be included in the enum.
    - 2798 – The schemas shall indicate if an Property is read only or read-write.
      - 2799 – By means of the readOnly schema tag belonging to the Property.
      - 2800 – Default value of readOnly is false as defined by OpenAPI 2.0.
    - 2801 – The default value of the "rt" Property shall be used to indicate the supported
    - 2802 Resource Types.
    - 2803 – oneOf and anyOf constructs are allowed to be used as part of a OpenAPI 2.0 schema
    - 2804 object. The OpenAPI 2.0 schema with oneOf and anyOf constructs can be found in
    - 2805 Annex **B.1**.
  - 2806 – For Atomic Measurements (see clause 7.8.4), the following apply:
    - 2807 – The "rts" Property Value in the IDD shall include only the Resource Types the instance
    - 2808 contains and not the theoretical maximal set allowed by the schema definition.
    - 2809 – The Resources that are part of an Atomic Measurement, excluding the Atomic Measurement
    - 2810 Resource itself, shall not be added to their own individual path in the IDD, as they are not
    - 2811 individually addressable; however, the schemas for the composed Resource Types shall be
    - 2812 provided in the IDD as part of the batch response definition along with the "href" for the
    - 2813 Resource.
- 2814 Dynamic Resources (e.g. Resources that can be created on a request by a Client) shall have a
- 2815 URL definition which contains a URL identifier (e.g. using the {} syntax). A URL with {} identifies
- 2816 that the Resource definition applies to the whole group of Resources that may be created. The
- 2817 actual path may contain the Collection node that links to the Resource.

2818 Example of a URL with identifiers:

2819 /SceneListResURI/{SceneCollectionResURI}/{SceneMemberResURI}:

2820 When different Resource Types are allowed to be created in a Collection, then the different

2821 schemas for the CREATE method shall define all possible Resource Types that may be created.

2822 The schema construct oneOf allows the definition of a schema with selectable Resources. The

2823 oneOf construct allows the integration of all schemas and that only one existing sub schema shall

2824 be used to indicate the definition of the Resource that may be created.

2825 Example usage of oneOf JSON schema construct is shown in Figure 12:

```

2826 {
2827   "oneOf": [
2828     { <<subschema 1 definition>> },
2829     { << sub schema 2 definition >> }
2830     ...
2831   ]
2832 }

```

2833 **Figure 12 – Example usage of oneOf JSON schema**

2834 A Client using the IDD of a Device should check the version of the supported IDD of the Device.

2835 The OpenAPI 2.0 version is indicated in each file with the tag "swagger". Example of the 2.0

Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved 77

2836 supported version of the tag is: "swagger": "2.0". Later versions of this document may reference  
 2837 newer versions of the OpenAPI specification, for example 3.0.

2838 A Device shall support one Resource with a Resource Type of "oic.wk.introspection" as defined in  
 2839 Table 25. The Resource with a Resource Type of "oic.wk.introspection" shall be included in the  
 2840 Resource "/oic/res".

2841 An empty IDD file, e.g. no URLs are exposed, shall still have the mandatory OpenAPI 2.0 fields.  
 2842 See OpenAPI specification. An example of an empty OpenAPI 2.0 file can be found in found in  
 2843 Annex B.2.

2844 **Table 25 – Introspection Resource**

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

2845

2846 Table 26 defines "oic.wk.introspection" Resource Type.

2847 **Table 26 – "oic.wk.introspection" Resource Type definition**

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

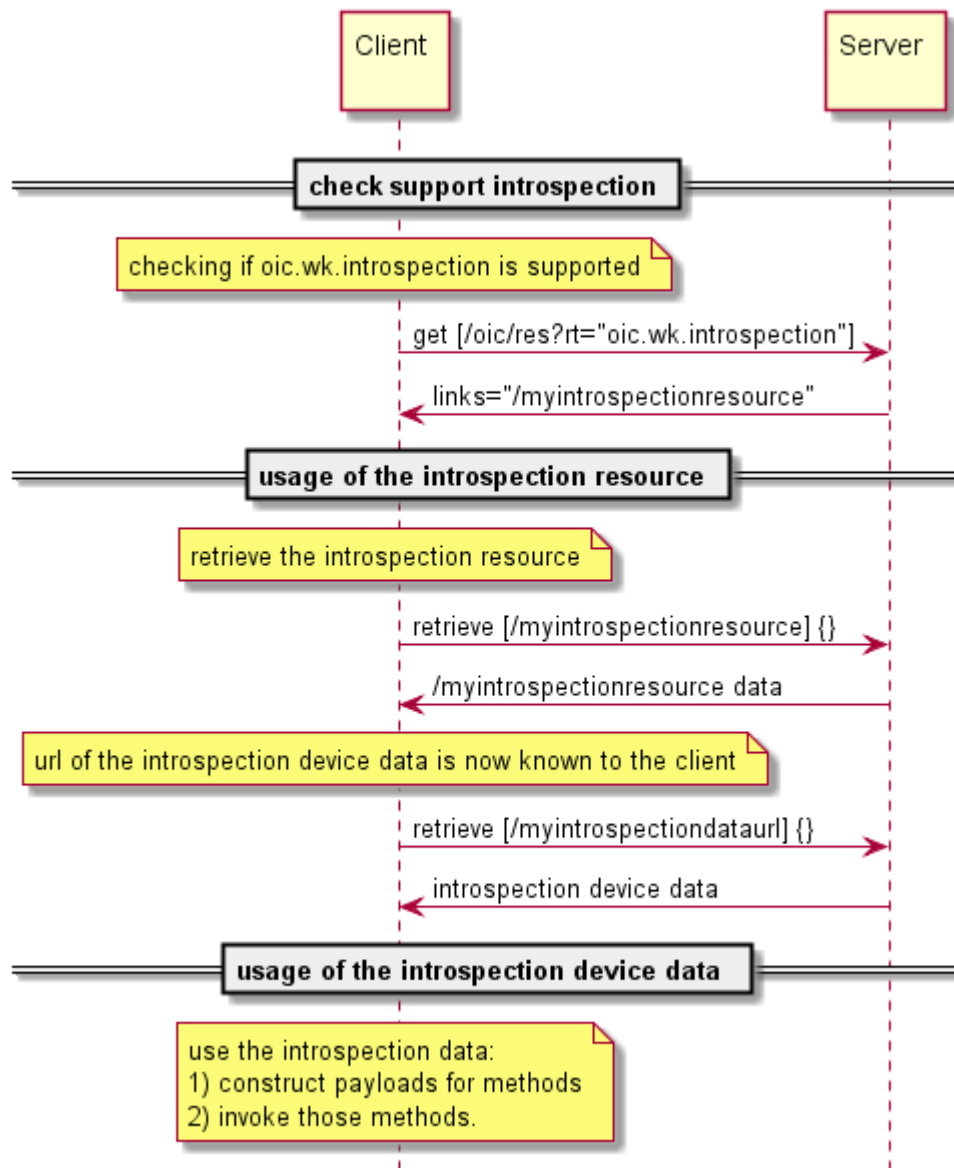
2848

#### 2849 **11.4.2 Usage of Introspection**

2850 The Introspection Device Data is retrieved in the following steps and as depicted in Figure 13:

- 2851 – Check if the Introspection Resource is supported and retrieve the URL of the Resource.
- 2852 – Retrieve the contents of the Introspection Resource
- 2853 – Download the Introspection Device Data from the URL specified the Introspection Resource.
- 2854 – Usage of the Introspection Device Data by the Client

2855



2856

2857 **Figure 13 – Interactions to check Introspection support and download the Introspection**  
 2858 **Device Data.**

2859 **12 Messaging**

2860 **12.1 Introduction**

2861 This clause specifies the protocol messaging mapping to the CRUDN messaging operations (clause  
 2862 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols is expected  
 2863 in later version of this document. All the Property information from the Resource model shall be  
 2864 carried within the message payload. This payload shall be generated in the Resource model layer  
 2865 and shall be encapsulated in the data connectivity layer. The message header shall only be used  
 2866 to describe the message payload (e.g., verb, mime-type, message payload format), in addition to  
 2867 the mandatory header fields defined in a messaging protocol (e.g., CoAP) specification. If the  
 2868 message header does not support this, then this information shall also be carried in the message  
 2869 payload. Resource model information shall not be included in the message header structure unless  
 2870 the message header field is mandatory in the messaging protocol specification.



2871 When a Resource is specified with a RESTful description language like OpenAPI 2.0 then the HTTP  
 2872 syntax definitions are used in the description (e.g., HTTP syntax for the CRUDN operations, status  
 2873 codes, etc). The HTTP syntax will be mapped to the actual used web transfer protocol (e.g., CoAP).

2874 **12.2 Mapping of CRUDN to CoAP**

2875 **12.2.1 Overview**

2876 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in clause  
 2877 12.2.3. A Device implementing CoAP shall conform to IETF RFC 7641 to implement the CoAP  
 2878 Observe option. Support for CoAP block transfer when the payload is larger than the MTU is defined  
 2879 in 12.2.8.

2880 **12.2.2 URIs**

2881 An OCF: URI is mapped to a coap: URI by replacing the scheme name "ocf" with "coap" if unsecure  
 2882 or "coaps" if secure before sending over the network by the requestor. Similarly on the receiver  
 2883 side, the scheme name is replaced with "ocf".

2884 Any query string that is present within the URI is encoded as one or more URI-Query Options as  
 2885 defined in IETF RFC 7252 clause 6.4.

2886 **12.2.3 CoAP method with request and response**

2887 **12.2.3.1 Overview**

2888 Every request has a CoAP method that realizes the request. The primary methods and their  
 2889 meanings are shown in Table 27, which provides the mapping of GET/POST/DELETE methods to  
 2890 CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides the generic  
 2891 behaviours when using these methods, however Resource OCF Interfaces may modify these  
 2892 generic semantics. The HTTP codes in the RESTful descriptions will be translated as described in  
 2893 IETF RFC 8075 clause 7 Response Code Mapping. CoAP methods not listed in Table 27 are not  
 2894 supported.

2895 **Table 27 – CoAP request and response**

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

2896

2897

2898 **12.2.3.2 CREATE with POST**

2899 POST shall be used only in situations where the request URI is valid, that is it is the URI of an  
2900 existing Resource on the Server that is processing the request. If no such Resource is present, the  
2901 Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall use  
2902 an existing request URI which identifies the Resource on the Server responsible for creation. The  
2903 URI of the created Resource is determined by the Server and provided to the Client in the response.

2904 A Client shall include the representation of the new Resource in the request payload. The new  
2905 resource representation in the payload shall have all the necessary Properties to create a valid  
2906 Resource instance, i.e. the created Resource should be able to properly respond to the valid  
2907 Request with mandatory OCF Interface (e.g., "GET with ?if=oc.if.baseline").

2908 Upon receiving the POST request, the Server shall either:

- 2909 – Create the new Resource with a new URI, respond with the new URI for the newly created  
2910 Resource and a success response code (2.xx); or
- 2911 – respond with an error response code (4.xx or 5.xx).

2912 **12.2.3.3 RETRIEVE with GET**

2913 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of  
2914 the target Resource identified by the request URI.

2915 Upon receiving the GET request, the Server shall either:

- 2916 – Send back the response with the representation of the target Resource with a success response  
2917 code (2.xx); or
- 2918 – respond with an error response code (4.xx or 5.xx) or ignore it (e.g. non-applicable multicast  
2919 GET).

2920 GET is a safe method and is idempotent.

2921 **12.2.3.4 UPDATE with POST**

2922 POST shall be used only in situations where the request URI is valid, that is it is the URI of an  
2923 existing Resource on the Server that is processing the request. If no such Resource is present, the  
2924 Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE  
2925 Property values of an existing Resource.

2926 Upon receiving the request, the Server shall either:

- 2927 – Apply the request to the Resource identified by the request URI in accordance with the applied  
2928 OCF Interface (i.e. POST for non-existent Properties is ignored) and send back a response with  
2929 a success response code (2.xx); or
- 2930 – respond with an error response code (4.xx or 5.xx). Note that if the representation in the payload  
2931 is incompatible with the target Resource for POST using the applied OCF Interface (i.e. the  
2932 overwrite semantic cannot be honored because of read-only Property in the payload), then the  
2933 error response code 4.xx shall be returned.

2934 **12.2.3.5 DELETE with DELETE**

2935 DELETE shall be used for DELETE operation. The DELETE method requests that the Resource  
2936 identified by the request URI be deleted.

2937 Upon receiving the DELETE request, the Server shall either:

- 2938 – Delete the target Resource and send back a response with a success response code (2.xx); or
- 2939 – respond with an error response code (4.xx or 5.xx).

2940 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

#### 2941 **12.2.4 Content-Format negotiation**

2942 The Framework mandates support of CBOR, however it allows for negotiation of the payload body  
2943 if more than one Content-Format (e.g. CBOR and JSON) is supported by an implementation. In this  
2944 case the Accept Option defined in clause 5.10.4 of IETF RFC 7252 shall be used to indicate which  
2945 Content-Format (e.g. JSON) is requested by the Client.

2946 The Content-Formats supported are shown in Table 28.

2947 **Table 28 – OCF Content-Formats**

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

2948  
2949 Clients shall include a Content-Format Option in every message that contains a payload. Servers  
2950 shall include a Content-Format Option for all success (2.xx) responses with a payload body. Per  
2951 IETF RFC 7252 clause 5.5.1, Servers shall include a Content-Format Option for all error (4.xx or  
2952 5.xx) responses with a payload body unless they include a Diagnostic Payload; error responses  
2953 with a Diagnostic Payload do not include a Content-Format Option. The Content-Format Option  
2954 shall use the ID column numeric value from Table 28. An OCF vertical may mandate a specific  
2955 Content-Format Option.

2956 Clients shall also include an Accept Option in every request message. The Accept Option shall  
2957 indicate the required Content-Format as defined in Table 28 for response messages. The Server  
2958 shall return the required Content-Format if available. If the required Content-Format cannot be  
2959 returned, then the Server shall respond with an appropriate error message.

#### 2960 **12.2.5 OCF-Content-Format-Version information**

2961 Servers and Clients shall include the OCF-Content-Format-Version Option in both request and  
2962 response messages with a payload. Clients shall include the OCF-Accept-Content-Format-Version  
2963 Option in request messages. The OCF-Content-Format-Version Option and OCF-Accept-Content-  
2964 Format-Version Option are specified as Option Numbers in the CoAP header as shown in Table 29.

2965 **Table 29 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option**  
2966 **Numbers**

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

2967  
2968 The value of both the OCF-Accept-Content-Format-Version Option and the OCF-Content-Format-  
2969 Version Option is a two-byte unsigned integer that is used to define the major, minor and sub  
2970 versions. The major and minor versions are represented by 5 bits and the sub version is  
2971 represented by 6 bits as shown in Table 30.

2972  
2973

**Table 30 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version Representation**

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

2974

2975 Table 31 illustrates several examples:

2976  
2977

**Table 31 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Representation**

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

2978

2979 The OCF-Accept-Content-Format-Version Option and OCF-Content-Format-Version Option for this  
2980 version of the document shall be "1.0.0" (i.e. "0b0000 1000 0000 0000").

2981 **12.2.6 Content-Format policy**

2982 All Devices shall support the current Content-Format Option, "application/vnd.ocf+cbor", and OCF-  
2983 Content-Format-Version "1.0.0".

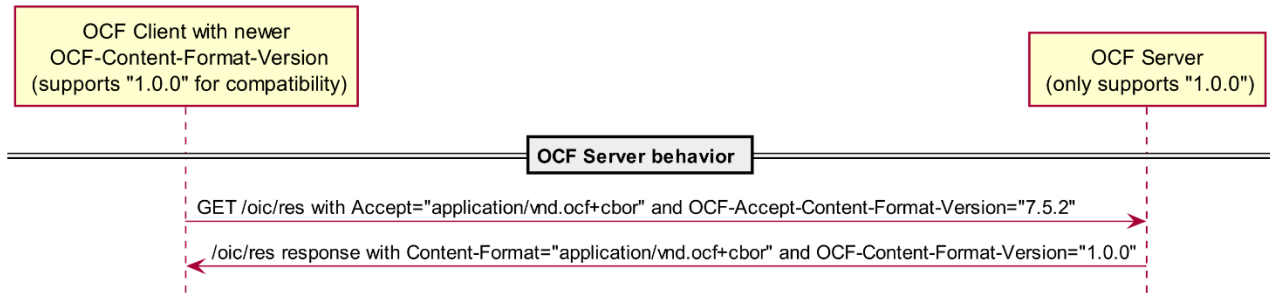
2984 For backward compatibility with previous OCF-Content-Format-Version Options:

- 2985 – All Client Devices shall support OCF-Content-Format-Version Option set to "1.0.0" and higher.
- 2986 – All Client Devices shall support OCF-Accept-Content-Format-Version Option set to "1.0.0" and  
2987 higher.
- 2988 – A Client shall send a discovery request message with its Accept Option set to  
2989 "application/vnd.ocf+cbor", and its OCF-Accept-Content-Format-Version Option matching its  
2990 highest supported version.
- 2991 – A Server shall respond to a Client's discovery request that is higher than its OCF-Content-  
2992 Format-Version by responding with its Content-Format Option set to "application/vnd.ocf+cbor",  
2993 and OCF-Content-Format-Version matching its highest supported version. The response  
2994 representation shall be encoded with the OCF-Content-Format-Version matching the Server's  
2995 highest supported version.
- 2996 – A Server may support previous Content-Formats and OCF-Content-Format-Versions to support  
2997 backward compatibility with previous versions.
- 2998 – For a Server that supports multiple OCF-Content-Format-Version Options, the Server should  
2999 attempt to respond with an OCF-Content-Format-Version that matches the OCF-Accept-  
3000 Content-Format-Version of the request.

3001 To maintain compatibility between Devices implemented to different versions of this document,  
3002 Devices should follow the policy as described in Figure 14.

3003 The OCF Clients in Figure 14 support sending Content-Format Option set to  
3004 "application/vnd.ocf+cbor", Accept Option set to "application/vnd.ocf+cbor", OCF-Content-Format-  
3005 Version Option set to "1.0.0", and OCF-Accept-Content-Format-Version Option set to "1.0.0"  
3006 (representing OCF 1.0 and later Clients). The OCF Servers in Figure 14 support sending Content-  
3007 Format Option set to "application/vnd.ocf+cbor" and OCF-Content-Format-Version Option set to  
3008 "1.0.0" (representing OCF 1.0 and later Servers).

3009



3010

3011 **Figure 14 – Content-Format Policy for backward compatible OCF Clients negotiating lower**  
3012 **OCF Content-Format-Version**

### 3013 12.2.7 CRUDN to CoAP response codes

3014 The mapping of CRUDN operations response codes to CoAP response codes are identical to the  
3015 response codes defined in IETF RFC 7252.

### 3016 12.2.8 CoAP block transfer

3017 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT  
3018 devices. However scenarios can be envisioned in which an application needs to transfer larger  
3019 payloads.

3020 CoAP block-wise transfer as defined in IETF RFC 7959 shall be used by all Servers which generate  
3021 a content payload that would exceed the size of a CoAP datagram as the result of handling any  
3022 defined CRUDN operation.

3023 Similarly, CoAP block-wise transfer as defined in IETF RFC 7959 shall be supported by all Clients.  
3024 The use of block-wise transfer is applied to both the reception of payloads as well as transmission  
3025 of payloads that would exceed the size of a CoAP datagram.

3026 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the  
3027 same reliability setting (i.e. all confirmable or all non-confirmable).

3028 A Client may support both the block1 (as descriptive) and block2 (as control) options as described  
3029 by IETF RFC 7959. A Server may support both the block1 (as control) and block2 (as descriptive)  
3030 options as described by IETF RFC 7959.

### 3031 12.2.9 Generic requirements for CoAP multicast

3032 A Client may use CoAP multicast to retrieve a target Resource with a fixed local path from multiple  
3033 other Devices. This clause provides generic requirements for this mechanism.

3034 – Devices shall join the All OCF Nodes multicast groups (as defined in [IANA IPv6 Multicast  
3035 Address Space Registry]) with scopes 2, 3, and 5 (i.e., ff02::158, ff03::158 and ff05::158) and  
3036 shall listen on the port 5683. For compliance to IETF RFC 7252 a Device may additionally join  
3037 the All CoAP Nodes multicast groups.

3038 – Clients intending to discover Resources shall join the multicast groups as defined in the first  
3039 bullet.

3040 – Clients shall send multicast requests to the All OCF Nodes multicast group address with scope  
3041 2 ("ff02::158") at port "5683". The requested URI shall be the fixed local path of the target  
3042 Resource optionally followed by query parameters. For compliance to IETF RFC 7252 a Client  
3043 may additionally send to the All CoAP Nodes multicast groups.

- 3044 – The multicast request shall be permitted by matching the request to an ACE which permits  
3045 unauthenticated access to the target Resource as described in ISO/IEC 30118-2:2018.
- 3046 – Handling of multicast requests shall be as described in clause 8 of IETF RFC 7252 and clause  
3047 4.1 in IETF RFC 6690.
- 3048 – Devices which receive the request shall respond, subject to query parameter processing  
3049 specific to the requested Resource.

## 3050 **12.3 Mapping of CRUDN to CoAP serialization over TCP**

### 3051 **12.3.1 Overview**

3052 In environments where TCP is already available, CoAP can take advantage of it to provide reliability.  
3053 Also in some environments UDP traffic is blocked, so deployments may use TCP. For example,  
3054 consider a cloud application acting as a Client and the Server is located at the user's home. A  
3055 Server which already support CoAP as a messaging protocol could easily support CoAP  
3056 serialization over TCP rather than utilizing another messaging protocol. A Device implementing  
3057 CoAP Serialization over TCP shall conform to IETF RFC 8323.

### 3058 **12.3.2 URIs**

3059 When UDP is blocked, Clients are dependent on pre-configured details of the Device to determine  
3060 if the Device supports CoAP serialization over TCP. When UDP is not-blocked, a Device which  
3061 supports CoAP serialization over TCP shall populate the "eps" Parameter in the "/oic/res" response,  
3062 as defined in 10.2, with the URI scheme(s) as defined in clause 8.1 or 8.2 of IETF RFC 8323. For  
3063 the "coaps+tcp" URI scheme, as defined in clause 8.2 of IETF RFC 8323, IETF RFC 7301 shall be  
3064 used. In addition, the URIs used for CoAP serialization over TCP shall conform to 12.2.2 by  
3065 substituting the scheme names with the scheme names defined in clauses 8.1 and 8.2 of  
3066 IETF RFC 8323 respectively.

### 3067 **12.3.3 CoAP method with request and response**

3068 The CoAP methods used for CoAP serialization over TCP shall conform to 12.2.3.

### 3069 **12.3.4 Content-Format negotiation**

3070 The Content Format negotiation used for CoAP serialization over TCP shall conform to 12.2.4.

### 3071 **12.3.5 OCF-Content-Format-Version information**

3072 The OCF Content Format Version information used for CoAP serialization over TCP shall conform  
3073 to 12.2.5.

### 3074 **12.3.6 Content-Format policy**

3075 The Content Format policy used for CoAP serialization over TCP shall conform to 12.2.6.

### 3076 **12.3.7 CRUDN to CoAP response codes**

3077 The CRUDN to CoAP response codes for CoAP serialization over TCP shall conform to 12.2.7.

### 3078 **12.3.8 CoAP block transfer**

3079 The CoAP block transfer for CoAP serialization over TCP shall conform to clause 6 of  
3080 IETF RFC 8323.

### 3081 **12.3.9 Keep alive (connection health)**

3082 The Device that initiated the CoAP over TCP connection shall send a Ping message as described  
3083 in clause 5.4 in IETF RFC 8323. The Device to which the connection was made may send a Ping  
3084 message. The recipient of any Ping message shall send a Pong message as described in clause  
3085 5.4 in IETF RFC 8323.

3086 Both sides of an established CoAP over TCP connection may send subsequent Ping (and  
3087 corresponding Pong) messages.

#### 3088 **12.4 Payload Encoding in CBOR**

3089 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to  
3090 JSON from CBOR in accordance with IETF RFC 7049 clause 4 unless otherwise specified in this  
3091 clause.

3092 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types  
3093 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-  
3094 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation  
3095 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer  
3096 numbers shall be within the closed interval  $[-2^{53}, 2^{53}]$ . Properties defined as a JSON number  
3097 should be encoded as integers whenever possible; if this is not possible Properties defined as a  
3098 JSON number should use single-precision if the loss of precision does not affect the quality of  
3099 service, otherwise the Property shall use double-precision.

3100 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values  
3101 in any position. If a Property defined as a JSON integer is received encoded other than as an  
3102 integer, the implementation may reject this encoding using a final response as appropriate for the  
3103 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a Property is  
3104 defined as a JSON number an implementation shall accept integers, single- and double-precision  
3105 floating point.

#### 3106 **13 Security**

3107 The details for handling security and privacy are specified in ISO/IEC 30118-2:2018.

3108  
3109  
3110  
3111

## Annex A (normative)

### Resource Type definitions

#### 3112 A.1 List of Resource Type definitions

3113 All the clauses in Annex A describe the Resource Types with a RESTful API definition language.  
3114 The Resource Type definitions presented in Annex A are formatted for readability, and so may  
3115 appear to have extra line breaks. Table A-1 contains the list of defined Core Common Resources  
3116 in this document.

3117 **Table A-1 – Alphabetized list of Core Resources**

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

3118

#### 3119 A.2 Atomic Measurement links list representation

##### 3120 A.2.1 Introduction

3121 The oic.if.baseline OCF Interface exposes a representation of the links and  
3122 the Common Properties of the Atomic Measurement Resource.  
3123

##### 3124 A.2.2 Example URI

3125 /AtomicMeasurementResURI

##### 3126 A.2.3 Resource type

3127 The Resource Type is defined as: "oic.wk.atomicmeasurement".

##### 3128 A.2.4 OpenAPI 2.0 definition

```
3129 {  
3130     "swagger": "2.0",  
3131     "info": {  
3132         "title": "Atomic Measurement links list representation",  
3133         "version": "2019-03-04",  
3134         "license": {  
3135             "name": "OCF Data Model License",  
3136             "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",  
3137             "x-copyright": "Copyright 2018-2019 Open Connectivity Foundation, Inc. All rights reserved."  
3138         }  
3139     },  
3140     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md",  
3141     "schemes": ["http"],  
3142     "consumes": ["application/json"],  
3143     "produces": ["application/json"],  
3144     "paths": {  
3145         "/AtomicMeasurementResURI?if=oic.if.ll": {  
3146             "get": {
```



```

3147         "description": "The oic.if.ll OCF Interface exposes a representation
3148 of the Links",
3149         "parameters": [
3150             {
3151                 "$ref": "#/parameters/interface-all"
3152             }
3153         ],
3154         "responses": {
3155             "200": {
3156                 "description": "",
3157                 "x-example": [{
3158                     "href": "/temperature",
3159                     "rt": ["oic.r.temperature"],
3160                     "if": ["oic.if.s", "oic.if.baseline"]
3161                 }],
3162                 {
3163                     "href": "/bodylocation",
3164                     "rt": ["oic.r.body.location.temperature"],
3165                     "if": ["oic.if.s", "oic.if.baseline"]
3166                 },
3167                 {
3168                     "href": "/timestamp",
3169                     "rt": ["oic.r.time.stamp"],
3170                     "if": ["oic.if.s", "oic.if.baseline"]
3171                 }
3172             ],
3173             "schema": {
3174                 "$ref": "#/definitions/links"
3175             }
3176         }
3177     },
3178     "/AtomicMeasurementResURI?if=oic.if.b": {
3179         "get": {
3180             "description": "The oic.if.b OCF Interface returns data items
3181 retrieved from Resources pointed to by the Links.\n",
3182             "parameters": [
3183                 {
3184                     "$ref": "#/parameters/interface-all"
3185                 }
3186             ],
3187             "responses": {
3188                 "200": {
3189                     "description": "Normal response, no errors, all
3190 Properties are returned correctly\n",
3191                     "x-example": [{
3192                         "href": "/temperature",
3193                         "rep": {
3194                             "temperature": 38,
3195                             "units": "C",
3196                             "range": [25, 45]
3197                         }
3198                     }],
3199                     {
3200                         "href": "/bodylocation",
3201                         "rep": {
3202                             "bloc": "ear"
3203                         }
3204                     },
3205                     {
3206                         "href": "/timestamp",
3207                         "rep": {
3208                             "timestamp": "2007-04-05T14:30+09:00"
3209                         }
3210                     }
3211             ],
3212             "schema": {
3213                 "$ref": "#/definitions/batch-retrieve"
3214             }
3215         }
3216     }
3217 }

```

```

3218     },
3219     "/AtomicMeasurementResURI?if=oic.if.baseline": {
3220         "get": {
3221             "description": "The oic.if.baseline OCF Interface exposes a
3222 representation of the links and\nthe Common Properties of the Atomic Measurement Resource.\n",
3223             "parameters": [
3224                 {
3225                     "$ref": "#/parameters/interface-all"
3226                 }
3227             ],
3228             "responses": {
3229                 "200": {
3230                     "description": "",
3231                     "x-example": {
3232                         "rt": ["oic.wk.atomicmeasurement"],
3233                         "if": ["oic.if.b", "oic.if.ll",
3234                             "oic.if.baseline"],
3235                         "rts": ["oic.r.temperature",
3236                             "oic.r.time.stamp"],
3237                         "rts-m": ["oic.r.temperature",
3238                             "oic.r.time.stamp"],
3239                         "links": [{
3240                             "href": "/temperature",
3241                             "rt": ["oic.r.temperature"],
3242                             "if": ["oic.if.s", "oic.if.baseline"]
3243                         },
3244                         {
3245                             "href": "/bodylocation",
3246                             "rt":
3247 ["oic.r.body.location.temperature"],
3248                             "if": ["oic.if.s", "oic.if.baseline"]
3249                         },
3250                         {
3251                             "href": "/timestamp",
3252                             "rt": ["oic.r.time.stamp"],
3253                             "if": ["oic.if.s", "oic.if.baseline"]
3254                         }
3255                     ],
3256                     "schema": {
3257                         "$ref": "#/definitions/baseline"
3258                     }
3259                 }
3260             }
3261         }
3262     },
3263     "parameters": {
3264         "interface-all": {
3265             "in": "query",
3266             "name": "if",
3267             "type": "string",
3268             "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
3269         }
3270     },
3271     "definitions": {
3272         "links": {
3273             "type": "array",
3274             "items": {
3275                 "$ref": "#/definitions/oic.oic-link"
3276             }
3277         },
3278         "batch-retrieve": {
3279             "title": "Collection Batch Retrieve Format (auto merged)",
3280             "minItems": 1,
3281             "items": {
3282                 "additionalProperties": true,
3283                 "properties": {
3284                     "href": {
3285                         "$ref":
3286 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3287 schema.json#/definitions/href"
3288

```

```

3289         },
3290         "rep": {
3291             "oneOf": [{
3292                 "description": "The response payload from a
3293 single Resource",
3294                 "type": "object"
3295             }],
3296             {
3297                 "description": " The response payload from a
3298 Collection (batch) Resource",
3299                 "items": {
3300                     "properties": {
3301                         "anchor": {
3302                             "$ref":
3303 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3304 schema.json#/definitions/anchor"
3305                         },
3306                         "di": {
3307                             "$ref":
3308 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3309 schema.json#/definitions/di"
3310                         },
3311                         "eps": {
3312                             "$ref":
3313 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3314 schema.json#/definitions/eps"
3315                         },
3316                         "href": {
3317                             "$ref":
3318 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3319 schema.json#/definitions/href"
3320                         },
3321                         "if": {
3322                             "description": "The OCF
3323 Interface set supported by this Resource",
3324                             "items": {
3325                                 "enum": [
3326                                     "oic.if.baseline",
3327                                     "oic.if.ll",
3328                                     "oic.if.b",
3329                                     "oic.if.rw",
3330                                     "oic.if.r",
3331                                     "oic.if.a",
3332                                     "oic.if.s"],
3333                                 "type":
3334 "string"
3335                             },
3336                             "minItems": 1,
3337                             "uniqueItems": true,
3338                             "type": "array"
3339                         },
3340                         "ins": {
3341                             "$ref":
3342 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3343 schema.json#/definitions/ins"
3344                         },
3345                         "p": {
3346                             "$ref":
3347 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3348 schema.json#/definitions/p"
3349                         },
3350                         "rel": {
3351                             "description": "The relation of the target URI
3352 referenced by the Link to the context URI",
3353                             "oneOf": [
3354                                 {
3355                                     "$ref":
3356 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3357 schema.json#/definitions/rel_array"
3358                                 }
3359                             ],

```

```

3360         {
3361             "$ref":
3362 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3363 schema.json#/definitions/rel_string"
3364         }
3365     ],
3366 },
3367         "rt": {
3368             "description":
3369 "Resource Type of the Resource",
3370             "items": {
3371                 "maxLength":
3372 64,
3373                 "type":
3374 "string"
3375             },
3376             "minItems": 1,
3377             "uniqueItems": true,
3378             "type": "array"
3379         },
3380         "title": {
3381             "$ref":
3382 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3383 schema.json#/definitions/title"
3384         },
3385         "type": {
3386             "$ref":
3387 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3388 schema.json#/definitions/type"
3389         }
3390     },
3391     "required": [
3392         "href",
3393         "rt",
3394         "if"
3395     ],
3396     "type": "object"
3397 },
3398 "type": "array"
3399     ]}
3400 }
3401 },
3402 "required": [
3403     "href",
3404     "rep"
3405 ],
3406 "type": "object"
3407 },
3408 "type": "array"
3409 },
3410 "baseline": {
3411     "properties": {
3412         "links": {
3413             "description": "A set of simple or individual Links.",
3414             "items": {
3415                 "$ref": "#/definitions/oic.oic-link"
3416             },
3417             "type": "array"
3418         },
3419         "n": { "$ref" :
3420 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3421 schema.json#/definitions/n"},
3422         "id": { "$ref" :
3423 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3424 schema.json#/definitions/id"},
3425         "rt": {
3426             "description": "Resource Type of this Resource",
3427             "items": {
3428                 "enum": ["oic.wk.atomicmeasurement"],
3429                 "type": "string",
3430                 "maxLength": 64

```

```

3431     },
3432     "minItems": 1,
3433     "readOnly": true,
3434     "uniqueItems": true,
3435     "type": "array"
3436   },
3437     "rts": {
3438       "description": "An array of Resource Types that are supported
3439 within an array of Links exposed by the Resource",
3440       "items": {
3441         "maxLength": 64,
3442         "type": "string"
3443       },
3444       "minItems": 1,
3445       "readOnly": true,
3446       "uniqueItems": true,
3447       "type": "array"
3448     },
3449     "rts-m": {
3450       "description": "An array of Resource Types that are mandatory
3451 to be exposed within an array of Links exposed by the Resource",
3452       "items": {
3453         "maxLength": 64,
3454         "type": "string"
3455       },
3456       "minItems": 1,
3457       "readOnly": true,
3458       "uniqueItems": true,
3459       "type": "array"
3460     },
3461     "if": {
3462       "description": "The OCF Interface set supported by this
3463 Resource",
3464       "items": {
3465         "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"],
3466         "type": "string"
3467       },
3468       "minItems": 3,
3469       "readOnly": true,
3470       "uniqueItems": true,
3471       "type": "array"
3472     }
3473   },
3474   "type": "object",
3475   "required": [
3476     "rt",
3477     "if",
3478     "links"
3479 ]
3480 },
3481 "oic.oic-link": {
3482   "properties": {
3483     "anchor": {
3484       "$ref":
3485 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3486 schema.json#/definitions/anchor"
3487     },
3488     "di": {
3489       "$ref":
3490 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3491 schema.json#/definitions/di"
3492     },
3493     "eps": {
3494       "$ref":
3495 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3496 schema.json#/definitions/eps"
3497     },
3498     "href": {
3499       "$ref":
3500 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3501 schema.json#/definitions/href"

```

```

3502     },
3503     "if": {
3504         "description": "The OCF Interface set supported by this
Resource",
3505     },
3506     "items": {
3507         "enum": [
3508             "oic.if.baseline",
3509             "oic.if.ll",
3510             "oic.if.b",
3511             "oic.if.rw",
3512             "oic.if.r",
3513             "oic.if.a",
3514             "oic.if.s"],
3515         "type": "string"
3516     },
3517     "minItems": 1,
3518     "uniqueItems": true,
3519     "type": "array"
3520 },
3521 "ins": {
3522     "$ref":
3523     "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/ins"
3524 },
3525 "p": {
3526     "$ref":
3527     "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/p"
3528 },
3529 "rel": {
3530     "description": "The relation of the target URI referenced by the Link to the context URI",
3531     "oneOf": [
3532         {
3533             "$ref":
3534             "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_array"
3535         },
3536         {
3537             "$ref":
3538             "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/rel_string"
3539         }
3540     ]
3541 },
3542 "rt": {
3543     "description": "Resource Type of the Resource",
3544     "items": {
3545         "maxLength": 64,
3546         "type": "string"
3547     },
3548     "minItems": 1,
3549     "uniqueItems": true,
3550     "type": "array"
3551 },
3552 "title": {
3553     "$ref":
3554     "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/title"
3555 },
3556 "type": {
3557     "$ref":
3558     "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
schema.json#/definitions/type"
3559 },
3560 "required": [
3561     "href",
3562     "rt",
3563     "if"
3564 ],
3565 "type": "object"

```

3573 }  
 3574 }  
 3575 }  
 3576 }

3577 **A.2.5 Property definition**

3578 Table A-2 defines the Properties that are part of the "oic.wk.atomicmeasurement" Resource Type.

3579 **Table A-2 – The Property definitions of the Resource with type "rt" =**  
 3580 **"oic.wk.atomicmeasurement".**

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

rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI.
rt	array: see schema	Yes	Read Write	Resource Type of the Resource.
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	

3581 **A.2.6 CRUDN behaviour**

3582 Table A-3 defines the CRUDN operations that are supported on the "oic.wk.atomicmeasurement"  
3583 Resource Type.

3584 **Table A-3 – The CRUDN operations of the Resource with type "rt" =**  
3585 **"oic.wk.atomicmeasurement".**

Create	Read	Update	Delete	Notify
	get			observe

3586 **A.3 Collection**

3587 **A.3.1 Introduction**

3588 Collection Resource Type contains Properties and Links.  
3589 The oic.if.baseline OCF Interface exposes a representation of  
3590 the Links and the Properties of the Collection Resource itself  
3591

3592 **A.3.2 Example URI**

3593 /CollectionResURI

3594 **A.3.3 Resource type**

3595 The Resource Type is defined as: "oic.wk.col".

3596 **A.3.4 OpenAPI 2.0 definition**

```
3597 {
3598   "swagger": "2.0",
3599   "info": {
3600     "title": "Collection",
3601     "version": "2019-03-04",
3602     "license": {
3603       "name": "OCF Data Model License",
3604       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
3605       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
3606     },
3607     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
3608   },
3609   "schemes": [
3610     "http"
3611   ],
3612   "consumes": [
3613     "application/json"
3614   ],
3615   "produces": [
3616     "application/json"
3617   ],
3618   "paths": {
```



```

3619     "/CollectionResURI?if=oic.if.ll" : {
3620       "get": {
3621         "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.ll OCF
3622 Interface exposes a representation of the Links\n",
3623         "parameters": [
3624           {
3625             "$ref": "#/parameters/interface-all"
3626           }
3627         ],
3628         "responses": {
3629           "200": {
3630             "description" : "",
3631             "x-example": [
3632               {
3633                 "href": "/switch",
3634                 "rt": ["oic.r.switch.binary"],
3635                 "if": ["oic.if.a", "oic.if.baseline"],
3636                 "eps": [
3637                   {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3638                   {"ep": "coaps://[fe80::b1d6]:1122"},
3639                   {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
3640                 ]
3641               },
3642               {
3643                 "href": "/airFlow",
3644                 "rt": ["oic.r.airflow"],
3645                 "if": ["oic.if.a", "oic.if.baseline"],
3646                 "eps": [
3647                   {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3648                   {"ep": "coaps://[fe80::b1d6]:1122"},
3649                   {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
3650                 ]
3651               }
3652             ],
3653             "schema": {
3654               "$ref": "#/definitions/slinks"
3655             }
3656           }
3657         }
3658       }
3659     },
3660     "/CollectionResURI?if=oic.if.baseline" : {
3661       "get": {
3662         "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.baseline
3663 OCF Interface exposes a representation of\nthe Links and the Properties of the Collection Resource
3664 itself\n",
3665         "parameters": [
3666           {
3667             "$ref": "#/parameters/interface-all"
3668           }
3669         ],
3670         "responses": {
3671           "200": {
3672             "description" : "",
3673             "x-example": {
3674               "rt": ["oic.wk.col"],
3675               "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
3676               "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
3677               "rts-m": [ "oic.r.switch.binary" ],
3678               "links": [
3679                 {
3680                   "href": "/switch",
3681                   "rt": ["oic.r.switch.binary"],
3682                   "if": ["oic.if.a", "oic.if.baseline"],
3683                   "eps": [
3684                     {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3685                     {"ep": "coaps://[fe80::b1d6]:1122"},
3686                     {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
3687                   ]
3688                 },
3689               ]
3690             }
3691           }
3692         }
3693       }
3694     }
3695   }

```

```

3690         "href": "/airFlow",
3691         "rt": ["oic.r.airflow"],
3692         "if": ["oic.if.a", "oic.if.baseline"],
3693         "eps": [
3694             {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
3695             {"ep": "coaps://[fe80::b1d6]:1122"},
3696             {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
3697         ]
3698     }
3699 ]
3700 },
3701 "schema": {
3702     "$ref": "#/definitions/sbaseline"
3703 }
3704 }
3705 }
3706 },
3707 "post": {
3708     "description": "Update on Baseline OCF Interface\n",
3709     "parameters": [
3710         {
3711             "$ref": "#/parameters/interface-update"
3712         },
3713         {
3714             "name": "body",
3715             "in": "body",
3716             "required": true,
3717             "schema": {
3718                 "$ref": "#/definitions/sbaseline-update"
3719             }
3720         }
3721     ],
3722     "responses": {
3723         "200": {
3724             "description": "",
3725             "schema": {
3726                 "$ref": "#/definitions/sbaseline"
3727             }
3728         }
3729     }
3730 },
3731 },
3732 "/CollectionResURI?if=oic.if.b" : {
3733     "get": {
3734         "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.b OCF
3735 Interface exposes a composite representation of the\nResources pointed to by the Links\n",
3736         "parameters": [
3737             {
3738                 "$ref": "#/parameters/interface-all"
3739             }
3740         ],
3741         "responses": {
3742             "200": {
3743                 "description": "All targets returned OK status",
3744                 "x-example": [
3745                     {
3746                         "href": "/switch",
3747                         "rep": {
3748                             "value": true
3749                         }
3750                     },
3751                     {
3752                         "href": "/airFlow",
3753                         "rep": {
3754                             "direction": "floor",
3755                             "speed": 3
3756                         }
3757                     }
3758                 ],
3759                 "schema": {
3760                     "$ref": "#/definitions/sbatch-retrieve"

```

```

3761     }
3762   },
3763   "404": {
3764     "description" : "One or more targets did not return an OK status, return a
3765 representation containing returned Properties from the targets that returned OK",
3766     "x-example": [
3767       {
3768         "href": "/switch",
3769         "rep": {
3770           "value": true
3771         }
3772       }
3773     ],
3774     "schema": {
3775       "$ref": "#/definitions/sbatch-retrieve"
3776     }
3777   }
3778 },
3779 "post": {
3780 "description": "Update on Batch OCF Interface\n",
3781 "parameters": [
3782   {
3783     "$ref": "#/parameters/interface-update"
3784   },
3785   {
3786     "name": "body",
3787     "in": "body",
3788     "required": true,
3789     "schema": {
3790       "$ref": "#/definitions/sbatch-update"
3791     },
3792     "x-example": [
3793       {
3794         "href": "/switch",
3795         "rep": {
3796           "value": true
3797         }
3798       },
3799       {
3800         "href": "/airFlow",
3801         "rep": {
3802           "direction": "floor",
3803           "speed": 3
3804         }
3805       }
3806     ]
3807   }
3808 ],
3809 "responses": {
3810   "200": {
3811     "description" : "All targets returned OK status, return a representation of the current
3812 state of all targets",
3813     "x-example": [
3814       {
3815         "href": "/switch",
3816         "rep": {
3817           "value": true
3818         }
3819       },
3820       {
3821         "href": "/airFlow",
3822         "rep": {
3823           "direction": "demist",
3824           "speed": 5
3825         }
3826       }
3827     ],
3828     "schema": {
3829       "$ref": "#/definitions/sbatch-retrieve"
3830     }
3831   }

```

```

3832     },
3833     "403": {
3834         "description" : "One or more targets did not return OK status; return a retrieve
3835 representation of the current state of all targets in the batch",
3836         "x-example": [
3837             {
3838                 "href": "/switch",
3839                 "rep": {
3840                     "value": true
3841                 }
3842             },
3843             {
3844                 "href": "/airFlow",
3845                 "rep": {
3846                     "direction": "floor",
3847                     "speed": 3
3848                 }
3849             }
3850         ],
3851         "schema": {
3852             "$ref": "#/definitions/sbatch-retrieve"
3853         }
3854     }
3855 }
3856 }
3857 },
3858 },
3859 "parameters": {
3860     "interface-all" : {
3861         "in" : "query",
3862         "name" : "if",
3863         "type" : "string",
3864         "enum" : ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
3865     },
3866     "interface-update" : {
3867         "in" : "query",
3868         "name" : "if",
3869         "type" : "string",
3870         "enum" : ["oic.if.b", "oic.if.baseline"]
3871     }
3872 },
3873 "definitions": {
3874     "sbaseline" : {
3875         "properties": {
3876             "links" : {
3877                 "description": "A set of simple or individual Links.",
3878                 "items": {
3879                     "$ref": "#/definitions/oic.oic-link"
3880                 },
3881                 "type": "array"
3882             },
3883             "n": {
3884                 "$ref" :
3885 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3886 schema.json#/definitions/n"
3887             },
3888             "id": {
3889                 "$ref" :
3890 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3891 schema.json#/definitions/id"
3892             },
3893             "rt": {
3894                 "$ref": "#/definitions/oic.core.rt-col"
3895             },
3896             "rts": {
3897                 "$ref": "#/definitions/oic.core.rt"
3898             },
3899             "rts-m": {
3900                 "$ref": "#/definitions/oic.core.rt"
3901             },
3902             "if": {

```

```

3903         "description": "The OCF Interfaces supported by this Resource",
3904         "items": {
3905             "enum": [
3906                 "oic.if.ll",
3907                 "oic.if.baseline",
3908                 "oic.if.b"
3909             ],
3910             "type": "string",
3911             "maxLength": 64
3912         },
3913         "minItems": 2,
3914         "uniqueItems": true,
3915         "readOnly": true,
3916         "type": "array"
3917     },
3918 },
3919 "additionalProperties": true,
3920 "type": "object",
3921 "required": [
3922     "rt",
3923     "if",
3924     "links"
3925 ],
3926 },
3927 "sbaseline-update": {
3928     "additionalProperties": true
3929 },
3930     "oic.core.rt-col": {
3931         "description": "Resource Type of the Resource",
3932         "items": {
3933             "enum": ["oic.wk.col"],
3934             "type": "string",
3935             "maxLength": 64
3936         },
3937         "minItems": 1,
3938         "uniqueItems": true,
3939         "readOnly": true,
3940         "type": "array"
3941     },
3942 "oic.core.rt": {
3943     "description": "Resource Type or set of Resource Types",
3944     "items": {
3945         "type": "string",
3946         "maxLength": 64
3947     },
3948     "minItems": 1,
3949     "uniqueItems": true,
3950     "readOnly": true,
3951     "type": "array"
3952 },
3953 "sbatch-retrieve" : {
3954     "minItems" : 1,
3955     "items" : {
3956         "additionalProperties": true,
3957         "properties": {
3958             "href": {
3959                 "$ref":
3960 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3961 schema.json#/definitions/href"
3962             },
3963             "rep": {
3964                 "oneOf": [
3965                     {
3966                         "description": "The response payload from a single Resource",
3967                         "type": "object"
3968                     },
3969                     {
3970                         "description": " The response payload from a Collection (batch) Resource",
3971                         "items": {
3972                             "$ref": "#/definitions/oic.oic-link"
3973                         }

```

```

3974         "type": "array"
3975     }
3976 ]
3977 }
3978 },
3979 "required": [
3980     "href",
3981     "rep"
3982 ],
3983 "type": "object"
3984 },
3985 "type" : "array"
3986 },
3987 "sbatch-update" : {
3988     "title" : "Collection Batch Update Format",
3989     "minItems" : 1,
3990     "items" : {
3991         "$ref": "#/definitions/sbatch-update.item"
3992     },
3993     "type" : "array"
3994 },
3995 "sbatch-update.item" : {
3996     "additionalProperties": true,
3997     "description": "Array of Resource representations to apply to the batch Collection, using href
3998 to indicate which Resource(s) in the batch to update. If the href Property is empty, effectively
3999 making the URI reference to the Collection itself, the representation is to be applied to all
4000 Resources in the batch",
4001     "properties": {
4002         "href": {
4003             "$ref":
4004 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4005 schema.json#/definitions/href"
4006         },
4007         "rep": {
4008             "oneOf": [
4009                 {
4010                     "description": "The payload for a single Resource",
4011                     "type": "object"
4012                 },
4013                 {
4014                     "description": " The payload for a Collection (batch) Resource",
4015                     "items": {
4016                         "$ref": "#/definitions/oic.oic-link"
4017                     },
4018                     "type": "array"
4019                 }
4020             ]
4021         }
4022     },
4023     "required": [
4024         "href",
4025         "rep"
4026     ],
4027     "type": "object"
4028 },
4029 "slinks" : {
4030     "type" : "array",
4031     "items" : {
4032         "$ref": "#/definitions/oic.oic-link"
4033     }
4034 },
4035 "oic.oic-link": {
4036     "properties": {
4037         "if": {
4038             "description": "The OCF Interfaces supported by the Linked target",
4039             "items": {
4040                 "enum": [
4041                     "oic.if.baseline",
4042                     "oic.if.ll",
4043                     "oic.if.b",
4044                     "oic.if.rw",

```

```

4045         "oic.if.r",
4046         "oic.if.a",
4047         "oic.if.s"
4048     ],
4049     "type": "string",
4050     "maxLength": 64
4051 },
4052 "minItems": 1,
4053 "uniqueItems": true,
4054 "readOnly": true,
4055 "type": "array"
4056 },
4057 "rt": {
4058     "$ref": "#/definitions/oic.core.rt"
4059 },
4060 "anchor": {
4061     "$ref":
4062 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4063 schema.json#/definitions/anchor"
4064 },
4065 "di": {
4066     "$ref":
4067 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4068 schema.json#/definitions/di"
4069 },
4070 "eps": {
4071     "$ref":
4072 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4073 schema.json#/definitions/eps"
4074 },
4075 "href": {
4076     "$ref":
4077 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4078 schema.json#/definitions/href"
4079 },
4080 "ins": {
4081     "$ref":
4082 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4083 schema.json#/definitions/ins"
4084 },
4085 "p": {
4086     "$ref":
4087 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4088 schema.json#/definitions/p"
4089 },
4090 "rel": {
4091     "$ref":
4092 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4093 schema.json#/definitions/rel_array"
4094 },
4095 "title": {
4096     "$ref":
4097 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4098 schema.json#/definitions/title"
4099 },
4100 "type": {
4101     "$ref":
4102 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4103 schema.json#/definitions/type"
4104 }
4105 },
4106 "required": [
4107     "href",
4108     "rt",
4109     "if"
4110 ],
4111 "type": "object"
4112 }
4113 }
4114 }
4115

```

4116 **A.3.5 Property definition**

4117 Table A-4 defines the Properties that are part of the "oic.wk.col" Resource Type.

4118 **Table A-4 – The Property definitions of the Resource with type "rt" = "oic.wk.col".**

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



4119 **A.3.6 CRUDN behaviour**

4120 Table A-5 defines the CRUDN operations that are supported on the "oic.wk.col" Resource Type.

4121 **Table A-5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col".**

Create	Read	Update	Delete	Notify
	get	post		observe

4122 **A.4 Device**

4123 **A.4.1 Introduction**

4124 Known Resource that is hosted by every Server.  
4125 Allows for logical Device specific information to be discovered.  
4126

4127 **A.4.2 Well-known URI**

4128 /oic/d

4129 **A.4.3 Resource type**

4130 The Resource Type is defined as: "oic.wk.d".

4131 **A.4.4 OpenAPI 2.0 definition**

```

4132 {
4133   "swagger": "2.0",
4134   "info": {
4135     "title": "Device",
4136     "version": "2019-03-13",
4137     "license": {
4138       "name": "OCF Data Model License",
4139       "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
4140       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4141     },
4142     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4143   },
4144   "schemes": [
4145     "http"
4146   ],
4147   "consumes": [
4148     "application/json"
4149   ],
4150   "produces": [
4151     "application/json"
4152   ],
4153   "paths": {
4154     "/oic/d" : {
4155       "get": {
4156         "description": "Known Resource that is hosted by every Server.\nAllows for logical Device
4157 specific information to be discovered.\n",
4158         "parameters": [
4159           {
4160             "$ref": "#/parameters/interface"
4161           }
4162         ],
4163         "responses": {
4164           "200": {
4165             "description": "",
4166             "x-example":
4167               {
4168                 "n": "Device 1",
4169                 "rt": ["oic.wk.d"],
4170                 "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
4171                 "icv": "ocf.2.0.2",
4172                 "dmv": "ocf.res.1.0.0, ocf.sh.1.0.0",

```

```

4173         "piid": "6F0AAC04-2BB0-468D-B57C-16570A26AE48"
4174     },
4175     "schema": {
4176         "$ref": "#/definitions/Device"
4177     }
4178 }
4179 }
4180 }
4181 },
4182 },
4183 "parameters": {
4184     "interface": {
4185         "in": "query",
4186         "name": "if",
4187         "type": "string",
4188         "enum": ["oic.if.r", "oic.if.baseline"]
4189     }
4190 },
4191 "definitions": {
4192     "Device": {
4193         "properties": {
4194             "rt": {
4195                 "description": "Resource Type of the Resource",
4196                 "items": {
4197                     "type": "string",
4198                     "maxLength": 64
4199                 },
4200                 "minItems": 1,
4201                 "readOnly": true,
4202                 "uniqueItems": true,
4203                 "type": "array"
4204             },
4205             "ld": {
4206                 "description": "Localized Descriptions.",
4207                 "items": {
4208                     "properties": {
4209                         "language": {
4210                             "allOf": [
4211                                 {
4212                                     "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4213 schema.json#/definitions/language-tag"
4214                                 },
4215                                 {
4216                                     "description": "An RFC 5646 language tag.",
4217                                     "readOnly": true
4218                                 }
4219                             ]
4220                         },
4221                         "value": {
4222                             "description": "Device description in the indicated language.",
4223                             "maxLength": 64,
4224                             "readOnly": true,
4225                             "type": "string"
4226                         }
4227                     },
4228                     "type": "object"
4229                 },
4230                 "minItems": 1,
4231                 "readOnly": true,
4232                 "type": "array"
4233             },
4234             "piid": {
4235                 "allOf": [
4236                     {
4237                         "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4238 schema.json#/definitions/uuid"
4239                     },
4240                     {
4241                         "description": "Protocol independent unique identifier for the Device that is
4242 immutable.",
4243                         "readOnly": true

```

```

4244     }
4245   ],
4246 },
4247 "di": {
4248   "allOf": [
4249     {
4250       "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4251 schema.json#/definitions/uuid"
4252     },
4253     {
4254       "description": "Unique identifier for the Device",
4255       "readOnly": true
4256     }
4257   ]
4258 },
4259 "dmno": {
4260   "description": "Model number as designated by manufacturer.",
4261   "maxLength": 64,
4262   "readOnly": true,
4263   "type": "string"
4264 },
4265 "sv": {
4266   "description": "Software version.",
4267   "maxLength": 64,
4268   "readOnly": true,
4269   "type": "string"
4270 },
4271 "dmn": {
4272   "description": "Manufacturer Name.",
4273   "items": {
4274     "properties": {
4275       "language": {
4276         "allOf": [
4277           {
4278             "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4279 schema.json#/definitions/language-tag"
4280           },
4281           {
4282             "description": "An RFC 5646 language tag.",
4283             "readOnly": true
4284           }
4285         ]
4286       },
4287       "value": {
4288         "description": "Manufacturer name in the indicated language.",
4289         "maxLength": 64,
4290         "readOnly": true,
4291         "type": "string"
4292       }
4293     },
4294     "type": "object"
4295   },
4296   "minItems": 1,
4297   "readOnly": true,
4298   "type": "array"
4299 },
4300 "icv": {
4301   "description": "The version of the Device",
4302   "maxLength": 64,
4303   "readOnly": true,
4304   "type": "string"
4305 },
4306 "dmv": {
4307   "description": "Specification versions of the Resource and Device Specifications to which
4308 this device data model is implemented",
4309   "maxLength": 256,
4310   "readOnly": true,
4311   "type": "string"
4312 },
4313 "n": {
4314   "$ref" :

```

```

4315 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4316 schema.json#/definitions/n"
4317 },
4318 "id": {
4319   "$ref" :
4320 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4321 schema.json#/definitions/id"
4322 },
4323 "if": {
4324   "description": "The OCF Interfaces supported by this Resource",
4325   "items": {
4326     "enum": [
4327       "oic.if.r",
4328       "oic.if.baseline"
4329     ],
4330     "type": "string",
4331     "maxLength": 64
4332   },
4333   "minItems": 2,
4334   "uniqueItems": true,
4335   "readOnly": true,
4336   "type": "array"
4337 },
4338 "econame" : {
4339   "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD.",
4340   "type": "string",
4341   "enum": ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"],
4342   "readOnly": true
4343 },
4344 "ecoversion" : {
4345   "description": "Version of ecosystem that a Bridged Device belongs to. Typical version
4346 string format is like n.n (e.g. 5.0).",
4347   "type": "string",
4348   "maxLength": 64,
4349   "readOnly": true
4350 },
4351 },
4352 "type": "object",
4353 "required": ["n", "di", "icv", "dmv", "piid"]
4354 }
4355 }
4356 }
4357

```

4358 **A.4.5 Property definition**

4359 Table A-6 defines the Properties that are part of the "oic.wk.d" Resource Type.

4360 **Table A-6 – The Property definitions of the Resource with type "rt" = "oic.wk.d".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
ld	array: see schema	No	Read Only	Localized Descriptions.
piid	multiple types: see schema	Yes	Read Write	
di	multiple types: see schema	Yes	Read Write	
dmno	string	No	Read Only	Model number as designated by manufacturer.
sv	string	No	Read Only	Software version.
dmn	array: see schema	No	Read Only	Manufacturer Name.

icv	string	Yes	Read Only	The version of the Device
dmv	string	Yes	Read Only	Specification versions of the Resource and Device Specifications to which this device data model is implemented.
n	multiple types: see schema	Yes	Read Write	
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfacces supported by this Resource.
econame	string	No	Read Only	Ecosystem Name of the Bridged Device which is exposed by this VOD.
ecoversion	string	No	Read Only	Version of ecosystem that a Bridged Device belongs to. Typical version string format is like n.n (e.g. 5.0).

4361 **A.4.6 CRUDN behaviour**

4362 Table A-7 defines the CRUDN operations that are supported on the "oic.wk.d" Resource Type.

4363 **Table A-7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d".**

Create	Read	Update	Delete	Notify
	get			observe

4364 **A.5 Introspection Resource**

4365 **A.5.1 Introduction**

4366 This Resource provides the means to get the Introspection Device Data (IDD) specifying all the  
 4367 OCF Endpoints of the Device.  
 4368 The url hosted by this Resource is either a local or an external url.  
 4369

4370 **A.5.2 Well-known URI**

4371 /IntrospectionResURI

4372 **A.5.3 Resource type**

4373 The Resource Type is defined as: "oic.wk.introspection".

4374 **A.5.4 OpenAPI 2.0 definition**

```

4375 {
4376   "swagger": "2.0",
4377   "info": {
4378     "title": "Introspection Resource",
4379     "version": "2019-03-04",
4380     "license": {

```

```

4381     "name": "OCF Data Model License",
4382     "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
4383     "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4384   },
4385   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4386 },
4387 "schemes": [
4388   "http"
4389 ],
4390 "consumes": [
4391   "application/json"
4392 ],
4393 "produces": [
4394   "application/json"
4395 ],
4396 "paths": {
4397   "/IntrospectionResURI": {
4398     "get": {
4399       "description": "This Resource provides the means to get the Introspection Device Data (IDD)
4400 specifying all the OCF Endpoints of the Device.\n\nThe url hosted by this Resource is either a local
4401 or an external url.\n\n",
4402       "parameters": [
4403         {
4404           "$ref": "#/parameters/interface"
4405         }
4406       ],
4407       "responses": {
4408         "200": {
4409           "description": "",
4410           "x-example": {
4411             "rt": ["oic.wk.introspection"],
4412             "urlInfo": [
4413               {
4414                 "content-type": "application/cbor",
4415                 "protocol": "coap",
4416                 "url": "coap://[fe80::1]:1234/IntrospectionExampleURI"
4417               }
4418             ]
4419           },
4420           "schema": {
4421             "$ref": "#/definitions/oic.wk.introspectionInfo"
4422           }
4423         }
4424       }
4425     }
4426   },
4427   "parameters": {
4428     "interface": {
4429       "in": "query",
4430       "name": "if",
4431       "type": "string",
4432       "enum": ["oic.if.r", "oic.if.baseline"]
4433     }
4434   },
4435   "definitions": {
4436     "oic.wk.introspectionInfo": {
4437       "properties": {
4438         "rt": {
4439           "description": "Resource Type of the Resource",
4440           "items": {
4441             "enum": ["oic.wk.introspection"],
4442             "type": "string",
4443             "maxLength": 64
4444           },
4445           "minItems": 1,
4446           "readOnly": true,
4447           "uniqueItems": true,
4448           "type": "array"
4449         },
4450         "n": {

```

```

4452         "$ref":
4453         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4454         schema.json#/definitions/n"
4455     },
4456     "urlInfo": {
4457         "description": "Information on the location of the Introspection Device Data (IDD).",
4458         "items": {
4459             "properties": {
4460                 "content-type": {
4461                     "default": "application/cbor",
4462                     "description": "content-type of the Introspection Device Data",
4463                     "enum": [
4464                         "application/json",
4465                         "application/cbor"
4466                     ],
4467                     "type": "string"
4468                 },
4469                 "protocol": {
4470                     "description": "Identifier for the protocol to be used to obtain the Introspection
4471 Device Data",
4472                     "enum": [
4473                         "coap",
4474                         "coaps",
4475                         "http",
4476                         "https",
4477                         "coap+tcp",
4478                         "coaps+tcp"
4479                     ],
4480                     "type": "string"
4481                 },
4482                 "url": {
4483                     "description": "The URL of the Introspection Device Data.",
4484                     "format": "uri",
4485                     "type": "string"
4486                 },
4487                 "version": {
4488                     "default": 1,
4489                     "description": "The version of the Introspection Device Data that can be
4490 downloaded",
4491                     "enum": [
4492                         1
4493                     ],
4494                     "type": "integer"
4495                 }
4496             },
4497             "required": [
4498                 "url",
4499                 "protocol"
4500             ],
4501             "type": "object"
4502         },
4503         "minItems": 1,
4504         "readOnly": true,
4505         "type": "array"
4506     },
4507     "id": {
4508         "$ref":
4509         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4510         schema.json#/definitions/id"
4511     },
4512     "if": {
4513         "description": "The OCF Interfaces supported by this Resource",
4514         "items": {
4515             "enum": [
4516                 "oic.if.r",
4517                 "oic.if.baseline"
4518             ],
4519             "type": "string",
4520             "maxLength": 64
4521         },
4522         "minItems": 2,

```

```

4523         "readOnly": true,
4524         "uniqueItems": true,
4525         "type": "array"
4526     }
4527 },
4528     "type" : "object",
4529     "required": ["urlInfo"]
4530 }
4531 }
4532 }
4533

```

4534 **A.5.5 Property definition**

4535 Table A-8 defines the Properties that are part of the "oic.wk.introspection" Resource Type.

4536 **Table A-8 – The Property definitions of the Resource with type "rt" =**  
4537 **"oic.wk.introspection".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	
urlInfo	array: see schema	Yes	Read Only	Information on the location of the Introspection Device Data (IDD).
id	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource.

4538 **A.5.6 CRUDN behaviour**

4539 Table A-9 defines the CRUDN operations that are supported on the "oic.wk.introspection" Resource  
4540 Type.

4541 **Table A-9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".**

Create	Read	Update	Delete	Notify
	get			observe

4542 **A.6 Platform**

4543 **A.6.1 Introduction**

4544 Known Resource that is defines the Platform on which an Server is hosted.  
4545 Allows for Platform specific information to be discovered.  
4546

4547 **A.6.2 Well-known URI**

4548 /oic/p

4549 **A.6.3 Resource type**

4550 The Resource Type is defined as: "oic.wk.p".



#### 4551 A.6.4 OpenAPI 2.0 definition

```
4552 {
4553   "swagger": "2.0",
4554   "info": {
4555     "title": "Platform",
4556     "version": "2019-03-04",
4557     "license": {
4558       "name": "OCF Data Model License",
4559       "url":
4560         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
4561 CENSE.md",
4562       "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4563     },
4564     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4565   },
4566   "schemes": ["http"],
4567   "consumes": ["application/json"],
4568   "produces": ["application/json"],
4569   "paths": {
4570     "/oic/p" : {
4571       "get": {
4572         "description": "Known Resource that is defines the Platform on which an Server is
4573 hosted.\nAllows for Platform specific information to be discovered.\n",
4574         "parameters": [
4575           {"$ref": "#/parameters/interface"}
4576         ],
4577         "responses": {
4578           "200": {
4579             "description" : "",
4580             "x-example": {
4581               "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
4582               "rt": ["oic.wk.p"],
4583               "mnmn": "Acme, Inc"
4584             },
4585             "schema": { "$ref": "#/definitions/Platform" }
4586           }
4587         }
4588       }
4589     }
4590   },
4591   "parameters": {
4592     "interface" : {
4593       "in" : "query",
4594       "name" : "if",
4595       "type" : "string",
4596       "enum" : ["oic.if.r", "oic.if.baseline"]
4597     }
4598   },
4599   "definitions": {
4600     "Platform" : {
4601       "properties": {
4602         "rt" : {
4603           "description": "Resource Type of the Resource",
4604           "items": {
4605             "enum": ["oic.wk.p"],
4606             "type": "string",
4607             "maxLength": 64
4608           },
4609           "minItems": 1,
4610           "uniqueItems": true,
4611           "readOnly": true,
4612           "type": "array"
4613         },
4614         "pi" : {
4615           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4616 9]{12}$",
4617           "type": "string",
4618           "description": "Platform Identifier",
4619           "readOnly": true
4620         }
4621       }
4622     }
4623   }
4624 }
```

```

4621     "mnfv" : {
4622         "description": "Manufacturer's firmware version",
4623         "maxLength": 64,
4624         "readOnly": true,
4625         "type": "string"
4626     },
4627     "vid" : {
4628         "description": "Manufacturer's defined information for the Platform. The content is
4629 freeform, with population rules up to the manufacturer",
4630         "maxLength": 64,
4631         "readOnly": true,
4632         "type": "string"
4633     },
4634     "mnmn" : {
4635         "description": "Manufacturer name",
4636         "maxLength": 64,
4637         "readOnly": true,
4638         "type": "string"
4639     },
4640     "mnmo" : {
4641         "description": "Model number as designated by the manufacturer",
4642         "maxLength": 64,
4643         "readOnly": true,
4644         "type": "string"
4645     },
4646     "mnhw" : {
4647         "description": "Platform Hardware Version",
4648         "maxLength": 64,
4649         "readOnly": true,
4650         "type": "string"
4651     },
4652     "mnos" : {
4653         "description": "Platform Resident OS Version",
4654         "maxLength": 64,
4655         "readOnly": true,
4656         "type": "string"
4657     },
4658     "mndt" : {
4659         "pattern": "^[0-9]{4})-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$",
4660         "type": "string",
4661         "description": "Manufacturing Date.",
4662         "readOnly": true
4663     },
4664     "id" : {
4665         "$ref":
4666 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4667 schema.json#/definitions/id"
4668     },
4669     "mns1" : {
4670         "description": "Manufacturer's Support Information URL",
4671         "format": "uri",
4672         "maxLength": 256,
4673         "readOnly": true,
4674         "type": "string"
4675     },
4676     "mnpv" : {
4677         "description": "Platform Version",
4678         "maxLength": 64,
4679         "readOnly": true,
4680         "type": "string"
4681     },
4682     "st" : {
4683         "description": "The date-time format pattern according to IETF RFC 3339.",
4684         "format": "date-time",
4685         "readOnly": true,
4686         "type": "string"
4687     },
4688     "n" : {
4689         "$ref":
4690 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4691 schema.json#/definitions/n"

```

```

4692     },
4693     "mnmml" : {
4694         "description": "Manufacturer's URL",
4695         "format": "uri",
4696         "maxLength": 256,
4697         "readOnly": true,
4698         "type": "string"
4699     },
4700     "mnsel" : {
4701         "description": "Serial number as designated by the manufacturer",
4702         "maxLength": 64,
4703         "readOnly": true,
4704         "type": "string"
4705     },
4706     "if" : {
4707         "description": "The OCF Interfaces supported by this Resource",
4708         "items": {
4709             "enum": [
4710                 "oic.if.r",
4711                 "oic.if.baseline"
4712             ],
4713             "type": "string",
4714             "maxLength": 64
4715         },
4716         "minItems": 2,
4717         "readOnly": true,
4718         "uniqueItems": true,
4719         "type": "array"
4720     },
4721     "mnct" : {
4722         "description": "An array of integers and each integer indicates the network connectivity
4723 type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-
4724 mib/ianaiftype-mib, e.g., [71, 259] which represents Wi-Fi and Zigbee.",
4725         "items": {
4726             "type": "integer",
4727             "minimum": 1,
4728             "description": "The network connectivity type based on IANAIfType value as defined by:
4729 https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib."
4730         },
4731         "minItems": 1,
4732         "readOnly": true,
4733         "type": "array"
4734     }
4735 },
4736 "type" : "object",
4737 "required": ["pi", "mnmml"]
4738 }
4739 }
4740 }
4741

```

#### 4742 A.6.5 Property definition

4743 Table A-10 defines the Properties that are part of the "oic.wk.p" Resource Type.

4744 **Table A-10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource.
pi	string	Yes	Read Only	Platform Identifier.
mnmml	string	No	Read Only	Manufacturer's firmware version.
vid	string	No	Read Only	Manufacturer's defined information for the Platform. The content is freeform, with population rules up to the manufacturer.

mnmn	string	Yes	Read Only	Manufacturer name.
mnmo	string	No	Read Only	Model number as designated by the manufacturer.
mnhw	string	No	Read Only	Platform Hardware Version.
mnos	string	No	Read Only	Platform Resident OS Version.
mnDt	string	No	Read Only	Manufacturing Date.
id	multiple types: see schema	No	Read Write	
mnsI	string	No	Read Only	Manufacturer's Support Information URL.
mnPv	string	No	Read Only	Platform Version
st	string	No	Read Only	The date-time format pattern according to IETF RFC 3339.
n	multiple types: see schema	No	Read Write	
mnml	string	No	Read Only	Manufacturer's URL.
mnsel	string	No	Read Only	Serial number as designated by the manufacturer.
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource.
mnNct	array: see schema	No	Read Only	An array of integers and each integer indicates the network connectivity type based on IANAIfType value as defined by: <a href="https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib">https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib</a> , e.g., [71, 259] which represents Wi-Fi and Zigbee.

4745 **A.6.6 CRUDN behaviour**

4746 Table A-11 defines the CRUDN operations that are supported on the "oic.wk.p" Resource Type.

4747 **Table A-11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p".**

Create	Read	Update	Delete	Notify
	get			observe

4748 **A.7 Discoverable Resources**

4749 **A.7.1 Introduction**

4750 Baseline representation of /oic/res; list of discoverable Resources  
4751

4752 **A.7.2 Well-known URI**

4753 /oic/res

4754 **A.7.3 Resource type**

4755 The Resource Type is defined as: "oic.wk.res".

4756 **A.7.4 OpenAPI 2.0 definition**

```
4757 {
4758   "swagger": "2.0",
4759   "info": {
4760     "title": "Discoverable Resources",
4761     "version": "2019-03-13",
4762     "license": {
```

```

4763     "name": "OCF Data Model License",
4764     "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
4765     "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
4766   },
4767   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4768 },
4769 "schemes": [
4770   "http"
4771 ],
4772 "consumes": [
4773   "application/json"
4774 ],
4775 "produces": [
4776   "application/json"
4777 ],
4778 "paths": {
4779   "/oic/res?if=oic.if.ll": {
4780     "get": {
4781       "description": "Links list representation of /oic/res; list of discoverable Resources\n",
4782       "parameters": [
4783         {
4784           "$ref": "#/parameters/interface-all"
4785         }
4786       ],
4787       "responses": {
4788         "200": {
4789           "description": "",
4790           "x-example": [
4791             {
4792               "href": "/humidity",
4793               "rt": ["oic.r.humidity"],
4794               "if": ["oic.if.s", "oic.if.baseline"],
4795               "p": {"bm": 3},
4796               "eps": [
4797                 {"ep": "coaps://[fe80::bld6]:1111", "pri": 2},
4798                 {"ep": "coaps://[fe80::bld6]:1122"},
4799                 {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
4800               ]
4801             }
4802           ],
4803           {
4804             "href": "/temperature",
4805             "rt": ["oic.r.temperature"],
4806             "if": ["oic.if.s", "oic.if.baseline"],
4807             "p": {"bm": 3},
4808             "eps": [
4809               {"ep": "coaps://[[2001:db8:a::123]:2222}
4810             ]
4811           }
4812         ],
4813         "schema": {
4814           "$ref": "#/definitions/slinklist"
4815         }
4816       }
4817     }
4818   },
4819   "/oic/res?if=oic.if.baseline": {
4820     "get": {
4821       "description": "Baseline representation of /oic/res; list of discoverable Resources\n",
4822       "parameters": [
4823         {
4824           "$ref": "#/parameters/interface-all"
4825         }
4826       ],
4827       "responses": {
4828         "200": {
4829           "description": "",
4830           "x-example": [
4831             {
4832               "rt": ["oic.wk.res"],
4833               "if": ["oic.if.ll", "oic.if.baseline"],

```

```

4834         "links": [
4835             {
4836                 "href": "/humidity",
4837                 "rt": ["oic.r.humidity"],
4838                 "if": ["oic.if.s", "oic.if.baseline"],
4839                 "p": {"bm": 3},
4840                 "eps": [
4841                     {"ep": "coaps://[fe80:b1d6]:1111", "pri": 2},
4842                     {"ep": "coaps://[fe80:b1d6]:1122"},
4843                     {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
4844                 ]
4845             },
4846             {
4847                 "href": "/temperature",
4848                 "rt": ["oic.r.temperature"],
4849                 "if": ["oic.if.s", "oic.if.baseline"],
4850                 "p": {"bm": 3},
4851                 "eps": [
4852                     {"ep": "coaps://[[2001:db8:a::123]:2222"}
4853                 ]
4854             }
4855         ]
4856     },
4857 ],
4858     "schema": {
4859         "$ref": "#/definitions/sbaseline"
4860     }
4861 }
4862 }
4863 }
4864 },
4865 },
4866 "parameters": {
4867     "interface-all": {
4868         "in": "query",
4869         "name": "if",
4870         "type": "string",
4871         "enum": ["oic.if.ll", "oic.if.baseline"]
4872     }
4873 },
4874 "definitions": {
4875     "oic.oic-link": {
4876         "type": "object",
4877         "properties": {
4878             "anchor": {
4879                 "$ref":
4880 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4881 schema.json#/definitions/anchor"
4882             },
4883             "di": {
4884                 "$ref":
4885 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4886 schema.json#/definitions/di"
4887             },
4888             "eps": {
4889                 "$ref":
4890 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4891 schema.json#/definitions/eps"
4892             },
4893             "href": {
4894                 "$ref":
4895 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4896 schema.json#/definitions/href"
4897             },
4898             "if": {
4899                 "description": "The OCF Interfaces supported by the Linked Resource",
4900                 "items": {
4901                     "enum": [
4902                         "oic.if.baseline",
4903                         "oic.if.ll",
4904                         "oic.if.b",

```

```

4905         "oic.if.rw",
4906         "oic.if.r",
4907         "oic.if.a",
4908         "oic.if.s"
4909     ],
4910     "type": "string",
4911     "maxLength": 64
4912 },
4913 "minItems": 1,
4914 "uniqueItems": true,
4915 "type": "array"
4916 },
4917 "ins": {
4918     "$ref":
4919 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4920 schema.json#/definitions/ins"
4921 },
4922 "p": {
4923     "$ref":
4924 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4925 schema.json#/definitions/p"
4926 },
4927 "rel": {
4928     "description": "The relation of the target URI referenced by the Link to the context URI",
4929     "oneOf": [
4930         {
4931             "$ref":
4932 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4933 schema.json#/definitions/rel_array"
4934         },
4935         {
4936             "$ref":
4937 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4938 schema.json#/definitions/rel_string"
4939         }
4940     ]
4941 },
4942 "rt": {
4943     "description": "Resource Type of the Linked Resource",
4944     "items": {
4945         "maxLength": 64,
4946         "type": "string"
4947     },
4948     "minItems": 1,
4949     "uniqueItems": true,
4950     "type": "array"
4951 },
4952 "title": {
4953     "$ref":
4954 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4955 schema.json#/definitions/title"
4956 },
4957 "type": {
4958     "$ref":
4959 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4960 schema.json#/definitions/type"
4961 }
4962 },
4963 "required": [
4964     "href",
4965     "rt",
4966     "if"
4967 ]
4968 },
4969 "slinklist": {
4970     "type": "array",
4971     "readOnly": true,
4972     "items": {
4973         "$ref": "#/definitions/oic.oic-link"
4974     }
4975 },

```

```

4976     "sbaseline": {
4977         "type": "array",
4978         "minItems": 1,
4979         "maxItems": 1,
4980         "items": {
4981             "type": "object",
4982             "properties": {
4983                 "n": {
4984                     "$ref":
4985 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4986 schema.json#/definitions/n"
4987                 },
4988                 "id": {
4989                     "$ref":
4990 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4991 schema.json#/definitions/id"
4992                 },
4993                 "rt": {
4994                     "description": "Resource Type of this Resource",
4995                     "items": {
4996                         "enum": ["oic.wk.res"],
4997                         "type": "string",
4998                         "maxLength": 64
4999                     },
5000                     "minItems": 1,
5001                     "readOnly": true,
5002                     "uniqueItems": true,
5003                     "type": "array"
5004                 },
5005                 "if": {
5006                     "description": "The OCF Interfaces supported by this Resource",
5007                     "items": {
5008                         "enum": [
5009                             "oic.if.ll",
5010                             "oic.if.baseline"
5011                         ],
5012                         "type": "string",
5013                         "maxLength": 64
5014                     },
5015                     "minItems": 2,
5016                     "readOnly": true,
5017                     "uniqueItems": true,
5018                     "type": "array"
5019                 },
5020                 "links": {
5021                     "type": "array",
5022                     "items": {
5023                         "$ref": "#/definitions/oic.oic-link"
5024                     }
5025                 }
5026             }
5027         },
5028         "required": [
5029             "rt",
5030             "if",
5031             "links"
5032         ]
5033     }
5034 }
5035 }
5036

```

### 5037 **A.7.5 Property definition**

5038 Table A-12 defines the Properties that are part of the "oic.wk.res" Resource Type.



**Table A-12 – The Property definitions of the Resource with type "rt" = "None".**

Property name	Value type	Mandatory	Access mode	Description
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the Linked Resource.
ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI.
rt	array: see schema	Yes	Read Write	Resource Type of the Linked Resource.
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource.
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource.
links	array: see schema	Yes	Read Write	

5040 **A.7.6 CRUDN behaviour**

5041 Table A-13 defines the CRUDN operations that are supported on the "None" Resource Type.

5042 **Table A-13 – The CRUDN operations of the Resource with type "rt" = "None".**

Create	Read	Update	Delete	Notify
	get			observe

5043  
5044  
5045  
5046

## Annex B (informative)

### OpenAPI 2.0 Schema Extension

5047

#### **B.1 OpenAPI 2.0 Schema Reference**

5048 OpenAPI 2.0 does not support allOf and anyOf JSON schema validation constructs; this document  
5049 has extended the underlying OpenAPI 2.0 schema to enable these, all OpenAPI 2.0 files are valid  
5050 against the extended schema. Reference the following location for a copy of the extended schema:

5051 <https://github.com/openconnectivityfoundation/OCFswagger2.0-schema>

5052

#### **B.2 OpenAPI 2.0 Introspection empty file**

5053 Reference the following location for a copy of an empty OpenAPI 2.0 file:

5054 [https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-  
5055 examples/introspection-empty.txt](https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-examples/introspection-empty.txt)  
5056