

# OCF Core Specification

VERSION 2.2.2 | February 2021



**OPEN** CONNECTIVITY  
FOUNDATION™

CONTACT [admin@openconnectivity.org](mailto:admin@openconnectivity.org)  
Copyright Open Connectivity Foundation, Inc. © 2021  
All Rights Reserved.

## Legal Disclaimer

NOTHING CONTAINED IN THIS DOCUMENT SHALL BE DEEMED AS GRANTING YOU ANY KIND OF LICENSE IN ITS CONTENT, EITHER EXPRESSLY OR IMPLIEDLY, OR TO ANY INTELLECTUAL PROPERTY OWNED OR CONTROLLED BY ANY OF THE AUTHORS OR DEVELOPERS OF THIS DOCUMENT. THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHORS AND DEVELOPERS OF THIS SPECIFICATION HEREBY DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OPEN 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 © 2016-2021 Open Connectivity Foundation, Inc. All rights reserved.

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

## CONTENTS

20		
21	Introduction .....	x
22	1 Scope.....	1
23	2 Normative references .....	1
24	3 Terms, definitions, and abbreviated terms .....	3
25	3.1 Terms and definitions .....	3
26	3.2 Symbols and abbreviated terms.....	7
27	4 Document conventions and organization .....	7
28	4.1 Conventions.....	7
29	4.2 Notation.....	7
30	4.3 Data types .....	8
31	4.4 Resource notation syntax .....	10
32	5 Architecture.....	10
33	5.1 Overview .....	10
34	5.2 Principle .....	11
35	5.3 Functional block diagram.....	12
36	5.4 Framework.....	13
37	6 Identification and addressing .....	14
38	6.1 Introduction.....	14
39	6.2 Identification .....	14
40	6.2.1 Device and Platform identification.....	14
41	6.2.2 Resource identification and addressing .....	14
42	6.3 Namespace:.....	16
43	6.4 Network addressing.....	16
44	7 Resource model .....	16
45	7.1 Introduction.....	16
46	7.2 Resource .....	17
47	7.3 Property.....	17
48	7.3.1 Introduction.....	17
49	7.3.2 Common Properties .....	18
50	7.4 Resource Type.....	19
51	7.4.1 Introduction.....	19
52	7.4.2 Resource Type Property.....	20
53	7.4.3 Resource Type definition.....	20
54	7.4.4 Multi-value "rt" Resource .....	22
55	7.5 Device Type.....	22
56	7.6 OCF Interface .....	23
57	7.6.1 Introduction.....	23
58	7.6.2 OCF Interface Property .....	23
59	7.6.3 OCF Interface methods .....	24
60	7.7 Resource representation .....	43
61	7.8 Structure.....	43
62	7.8.1 Introduction.....	43
63	7.8.2 Resource relationships (Links) .....	43

64	7.8.3	Collections.....	48
65	7.8.4	Atomic Measurement.....	50
66	7.9	Query Parameters.....	53
67	7.9.1	Introduction.....	53
68	7.9.2	Use of multiple parameters within a query .....	53
69	7.9.3	Application to multi-value "rt" Resources .....	53
70	7.9.4	OCF Interface specific considerations for queries .....	54
71	7.10	Error response payload .....	54
72	7.10.1	Overview .....	54
73	7.10.2	Error response payload content.....	54
74	7.10.3	Example of use .....	56
75	8	CRUDN .....	56
76	8.1	Overview .....	56
77	8.2	CREATE .....	57
78	8.2.1	Overview .....	57
79	8.2.2	CREATE request.....	57
80	8.2.3	Processing by the Server .....	58
81	8.2.4	CREATE response .....	58
82	8.3	RETRIEVE .....	58
83	8.3.1	Overview .....	58
84	8.3.2	RETRIEVE request .....	58
85	8.3.3	Processing by the Server .....	58
86	8.3.4	RETRIEVE response.....	59
87	8.4	UPDATE .....	59
88	8.4.1	Overview .....	59
89	8.4.2	UPDATE request.....	59
90	8.4.3	Processing by the Server .....	59
91	8.4.4	UPDATE response .....	60
92	8.5	DELETE.....	60
93	8.5.1	Overview .....	60
94	8.5.2	DELETE request .....	60
95	8.5.3	Processing by the Server .....	61
96	8.5.4	DELETE response.....	61
97	8.6	NOTIFY .....	61
98	8.6.1	Overview .....	61
99	8.6.2	NOTIFICATION response.....	61
100	9	Network and connectivity .....	61
101	9.1	Introduction.....	61
102	9.2	Architecture .....	62
103	9.3	IPv6 network layer requirements.....	63
104	9.3.1	Introduction.....	63
105	9.3.2	IPv6 node requirements .....	63
106	10	OCF Endpoint.....	63
107	10.1	OCF Endpoint definition .....	63
108	10.2	OCF Endpoint information .....	64

109	10.2.1	Introduction.....	64
110	10.2.2	"ep" .....	64
111	10.2.3	"pri" .....	65
112	10.2.4	"lat" .....	65
113	10.2.5	OCF Endpoint information in "eps" Parameter.....	65
114	10.3	OCF Endpoint discovery.....	66
115	10.3.1	Introduction.....	66
116	10.3.2	Implicit discovery .....	66
117	10.3.3	Explicit discovery with "/oic/res" response .....	67
118	11	Functional interactions.....	69
119	11.1	Introduction.....	69
120	11.2	Resource discovery.....	69
121	11.2.1	Introduction.....	69
122	11.2.2	Resource based discovery: mechanisms .....	69
123	11.2.3	Resource based discovery: Finding information .....	70
124	11.2.4	Resource discovery using "/oic/res" .....	77
125	11.2.5	Multicast discovery using "/oic/res" .....	78
126	11.2.6	Multicast discovery using "/.well-known/core" .....	79
127	11.3	Notification.....	79
128	11.3.1	Overview .....	79
129	11.3.2	Observe.....	80
130	11.4	Introspection.....	81
131	11.4.1	Overview .....	81
132	11.4.2	Usage of Introspection .....	84
133	11.5	Semantic Tags .....	85
134	11.5.1	Introduction.....	85
135	11.5.2	Semantic Tag definitions.....	86
136	12	Messaging.....	88
137	12.1	Introduction.....	88
138	12.2	Mapping of CRUDN to CoAP .....	89
139	12.2.1	Overview .....	89
140	12.2.2	URIs .....	89
141	12.2.3	CoAP method with request and response .....	89
142	12.2.4	Content-Format negotiation .....	91
143	12.2.5	OCF-Content-Format-Version information.....	91
144	12.2.6	Content-Format policy .....	92
145	12.2.7	CRUDN to CoAP response codes .....	93
146	12.2.8	CoAP block transfer .....	93
147	12.2.9	Generic requirements for CoAP multicast .....	94
148	12.2.10	Setting timeout on response to a confirmable request .....	94
149	12.2.11	Mapping the error response payload.....	94
150	12.3	Mapping of CRUDN to CoAP serialization over TCP.....	94
151	12.3.1	Overview .....	94
152	12.3.2	URIs .....	95
153	12.3.3	CoAP method with request and response .....	95

154	12.3.4	Content-Format negotiation .....	95
155	12.3.5	OCF-Content-Format-Version information .....	95
156	12.3.6	Content-Format policy .....	95
157	12.3.7	CRUDN to CoAP response codes .....	95
158	12.3.8	CoAP block transfer .....	95
159	12.3.9	Keep alive (connection health) .....	95
160	12.3.10	CoAP using a proxy .....	95
161	12.3.11	Mapping the error response payload .....	95
162	12.4	Payload Encoding in CBOR .....	96
163	13	Security .....	96
164	Annex A (normative)	Resource Type definitions .....	97
165	A.1	List of Resource Type definitions .....	97
166	A.2	Atomic Measurement links list representation .....	97
167	A.2.1	Introduction .....	97
168	A.2.2	Example URI .....	97
169	A.2.3	Resource type .....	97
170	A.2.4	OpenAPI 2.0 definition .....	97
171	A.2.5	Property definition .....	104
172	A.2.6	CRUDN behaviour .....	105
173	A.3	Collection .....	105
174	A.3.1	Introduction .....	105
175	A.3.2	Example URI .....	105
176	A.3.3	Resource type .....	105
177	A.3.4	OpenAPI 2.0 definition .....	105
178	A.3.5	Property definition .....	113
179	A.3.6	CRUDN behaviour .....	114
180	A.4	Device .....	114
181	A.4.1	Introduction .....	114
182	A.4.2	Well-known URI .....	114
183	A.4.3	Resource type .....	114
184	A.4.4	OpenAPI 2.0 definition .....	114
185	A.4.5	Property definition .....	117
186	A.4.6	CRUDN behaviour .....	118
187	A.5	Introspection Resource .....	119
188	A.5.1	Introduction .....	119
189	A.5.2	Well-known URI .....	119
190	A.5.3	Resource type .....	119
191	A.5.4	OpenAPI 2.0 definition .....	119
192	A.5.5	Property definition .....	121
193	A.5.6	CRUDN behaviour .....	121
194	A.6	Platform .....	122
195	A.6.1	Introduction .....	122
196	A.6.2	Well-known URI .....	122
197	A.6.3	Resource type .....	122
198	A.6.4	OpenAPI 2.0 definition .....	122

199	A.6.5	Property definition.....	125
200	A.6.6	CRUDN behaviour.....	125
201	A.7	Discoverable Resources.....	126
202	A.7.1	Introduction.....	126
203	A.7.2	Well-known URI .....	126
204	A.7.3	Resource type.....	126
205	A.7.4	OpenAPI 2.0 definition .....	126
206	A.7.5	Property definition.....	131
207	A.7.6	CRUDN behaviour.....	132
208	Annex B (informative)	OpenAPI 2.0 Schema Extension.....	133
209	B.1	OpenAPI 2.0 Schema Reference .....	133
210	B.2	OpenAPI 2.0 Introspection empty file .....	133
211	Annex C (normative)	Semantic Tag enumeration support.....	134
212	C.1	Introduction.....	134
213	C.2	"tag-pos-desc" supported enumeration .....	134
214	C.3	"tag-loc" supported enumeration .....	134
215	Bibliography	.....	136
216			
217			

## Figures

Figure 1 – Architecture - concepts.....	11
Figure 2 – Functional block diagram.....	12
Figure 3 – Communication layering model.....	13
Figure 4 – Example Resource .....	17
Figure 5 – CREATE operation .....	57
Figure 6 – RETRIEVE operation.....	58
Figure 7 – UPDATE operation .....	59
Figure 8 – DELETE operation.....	60
Figure 9 – High Level Network & Connectivity Architecture .....	62
Figure 10 – Resource based discovery: Finding information.....	70
Figure 11 – Observe Mechanism .....	80
Figure 12 – Example usage of oneOf JSON schema.....	83
Figure 13 – Interactions to check Introspection support and download the Introspection Device Data.....	85
Figure 14 – "tag-pos-rel" definition .....	87
Figure 15 – Content-Format Policy for backward compatible OCF Clients negotiating lower OCF Content-Format-Version.....	93
Figure C.1 – Enumeration for "tag-pos-desc" Semantic Tag .....	134
Figure C.2 – Definition of "tag-pos-desc" Semantic Tag values .....	134
Figure C.3 – Enumeration for "tag-loch" Semantic Tag.....	135



## Tables

242		
243		
244	Table 1 – Additional OCF Types.....	9
245	Table 2 – Name Property Definition.....	19
246	Table 3 – Resource Identity Property Definition .....	19
247	Table 4 – Resource Type Common Property definition.....	20
248	Table 5 – Example foobar Resource Type .....	21
249	Table 6 – Example foobar Properties.....	21
250	Table 7 – Resource Interface Property definition .....	23
251	Table 8 – OCF standard OCF Interfaces.....	24
252	Table 9 – Batch OCF Interface Example.....	31
253	Table 10 – Link target attributes list .....	45
254	Table 11 – "bm" Property definition .....	45
255	Table 12 – Resource Types Property definition.....	48
256	Table 13 – Mandatory Resource Types Property definition.....	48
257	Table 14 – Common Properties for Collections (in addition to Common Properties defined	
258	in 7.3.2).....	50
259	Table 15 – Common Properties for Atomic Measurement (in addition to Common	
260	Properties defined in 7.3.2) .....	51
261	Table 16 – Atomic Measurement Resource Type .....	52
262	Table 17 – Properties for Atomic Measurement (in addition to Common Properties defined	
263	in 7.3.2).....	52
264	Table 18 – Standardized error message .....	55
265	Table 18 – Parameters of CRUDN messages .....	57
266	Table 19 – "ep" value for Transport Protocol Suite.....	65
267	Table 20 – List of Core Resources .....	69
268	Table 21 – Mandatory discovery Core Resources .....	71
269	Table 22 – "oic.wk.res" Resource Type definition.....	72
270	Table 23 – Protocol scheme registry .....	73
271	Table 24 – "oic.wk.d" Resource Type definition .....	74
272	Table 25 – "oic.wk.p" Resource Type definition .....	76
273	Table 26 – Introspection Resource .....	84
274	Table 27 – "oic.wk.introspection" Resource Type definition.....	84
275	Table 28 – "tag-pos-desc" Semantic Tag definition .....	86
276	Table 29 – "tag-pos-rel" Semantic Tag definition .....	87
277	Table 30 – "tag-func-desc" Semantic Tag definition .....	88
278	Table 31 – "tag-locn" Semantic Tag definition.....	88
279	Table 31 – CoAP request and response .....	89
280	Table 32 – OCF Content-Formats.....	91
281	Table 33 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option	
282	Numbers.....	92

283	Table 34 – OCF-Accept-Content-Format-Version and OCF-Content-Format-Version	
284	Representation .....	92
285	Table 35 – Examples of OCF-Content-Format-Version and OCF-Accept-Content-Format-	
286	Version Representation.....	92
287	Table A.1 – Alphabetized list of Core Resources .....	97
288	Table A.2 – The Property definitions of the Resource with type "rt" =	
289	"oic.wk.atomicmeasurement".....	104
290	Table A.3 – The CRUDN operations of the Resource with type "rt" =	
291	"oic.wk.atomicmeasurement".....	105
292	Table A.4 – The Property definitions of the Resource with type "rt" = "oic.wk.col".....	113
293	Table A.5 – The CRUDN operations of the Resource with type "rt" = "oic.wk.col". .....	114
294	Table A.6 – The Property definitions of the Resource with type "rt" = "oic.wk.d". .....	118
295	Table A.7 – The CRUDN operations of the Resource with type "rt" = "oic.wk.d". .....	118
296	Table A.8 – The Property definitions of the Resource with type "rt" =	
297	"oic.wk.introspection".....	121
298	Table A.9 – The CRUDN operations of the Resource with type "rt" = "oic.wk.introspection".	122
299	Table A.10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".....	125
300	Table A.11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p". .....	125
301	Table A.12 – The Property definitions of the Resource with type "rt" = "oic.wk.res".....	131
302	Table A.13 – The CRUDN operations of the Resource with type "rt" = "oic.wk.res". .....	132
303		
304		

## Introduction

This document, and all the other parts associated with this document, were developed in response to worldwide demand for smart home focused Internet of Things (IoT) devices, such as appliances, door locks, security cameras, sensors, and actuators; these to be modelled and securely controlled, locally and remotely, over an IP network.

While some inter-device communication existed, no universal language had been developed for the IoT. Device makers instead had to choose between disparate frameworks, limiting their market share, or developing across multiple ecosystems, increasing their costs. The burden then falls on end users to determine whether the products they want are compatible with the ecosystem they bought into, or find ways to integrate their devices into their network, and try to solve interoperability issues on their own.

In addition to the smart home, IoT deployments in commercial environments are hampered by a lack of security. This issue can be avoided by having a secure IoT communication framework, which this standard solves.

The goal of these documents is then to connect the next 25 billion devices for the IoT, providing secure and reliable device discovery and connectivity across multiple OSs and platforms. There are multiple proposals and forums driving different approaches, but no single solution addresses the majority of key requirements. This document and the associated parts enable industry consolidation around a common, secure, interoperable approach.

## 1 Scope

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

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

## 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

393 IETF RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*, August  
394 2016  
395 <https://www.rfc-editor.org/info/rfc7959>

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

399 IETF RFC 8085, *UDP Usage Guidelines*, March 2017  
400 <https://www.rfc-editor.org/info/rfc8085>

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

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

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

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

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

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

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

416 **3 Terms, definitions, and abbreviated terms**

417 **3.1 Terms and definitions**

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

419 ISO and IEC maintain terminological databases for use in standardization at the following  
420 addresses:

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

423 **3.1.1**  
424 **Atomic Measurement**  
425 design pattern that ensures that the *Client* (3.1.6) can only access the *Properties* (3.1.34) of linked  
426 *Resources* (3.1.32) atomically, that is as a single group

427 **3.1.2**  
428 **Bridged Client**  
429 logical entity that accesses data via a *Bridged Protocol* (3.1.4)

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

431 **3.1.3**  
432 **Bridged Device**  
433 *Bridged Client* (3.1.2) or *Bridged Server* (3.1.5)

434 **3.1.4**  
435 **Bridged Protocol**  
436 another protocol (e.g., AllJoyn) that is being translated to or from OCF protocols

437 **3.1.5**  
438 **Bridged Server**  
439 logical entity that provides data via a *Bridged Protocol* (3.1.4)

440 Note 1 to entry: For example an AllJoyn Producer is a *Bridged Server* (3.1.5).  
441 Note 2 to entry: More than one *Bridged Server* (3.1.5) can exist on the same physical platform.

442 **3.1.6**  
443 **Client**  
444 logical entity that accesses a *Resource* (3.1.32) on a *Server* (3.1.37)

445 **3.1.7**  
446 **Collection**  
447 *Resource* (3.1.32) that contains zero or more *Links* (3.1.22)

448 **3.1.8**  
449 **Common Properties**  
450 *Properties* (3.1.34) specified for all *Resources* (3.1.32)

451 **3.1.9**  
452 **Composite Device**  
453 *Device* (3.1.13) that is modelled as multiple *Device Types* (3.1.14); with each component *Device*  
454 *Type* (3.1.14) being exposed as a *Collection* (3.1.7)

455 **3.1.10**  
456 **Configuration Source**  
457 cloud or service network or a local read-only file which contains and provides configuration related  
458 information to the *Devices* (3.1.13)

459 **3.1.11**  
460 **Core Resources**  
461 those *Resources* (3.1.32) that are defined in this document

462 **3.1.12**  
463 **Default OCF Interface**  
464 *OCF Interface* (3.1.19) used to generate the response when an *OCF Interface* (3.1.19) is omitted  
465 in a request

466 **3.1.13**  
467 **Device**  
468 logical entity that assumes one or more roles, e.g., *Client* (3.1.6), *Server* (3.1.37)

469 Note 1 to entry: More than one *Device* (3.1.13) can exist on a *Platform* (3.1.31).

470 **3.1.14**  
471 **Device Type**  
472 uniquely named definition indicating a minimum set of *Resource Types* (3.1.35) that a *Device*  
473 (3.1.13) supports

474 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  
475 use during *Resource* (3.1.32) discovery.

476 **3.1.15**  
477 **Device UUID**  
478 stack instance identifier

479 **3.1.16**  
480 **Discoverable Resource**  
481 *Resource* (3.1.32) that is listed in *"/oic/res"*

482 **3.1.17**  
483 **OCF Endpoint**  
484 entity participating in the OCF protocol, further identified as the source or destination of a request  
485 and response messages for a given Transport Protocol Suite

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

487 **3.1.18**  
488 **Framework**  
489 set of related functionalities and interactions defined in this document, which enable interoperability  
490 across a wide range of networked devices, including IoT

491 **3.1.19**  
 492 **OCF Interface**  
 493 interface description extended by OCF that provides a view to and permissible responses from a  
 494 *Resource* (3.1.32)

495 [SOURCE: IETF RFC 6690]

496 **3.1.20**  
 497 **Introspection**  
 498 mechanism to determine the capabilities of the hosted *Resources* (3.1.32) of a *Device* (3.1.13)

499 **3.1.21**  
 500 **Introspection Device Data (IDD)**  
 501 data that describes the payloads per implemented method of the *Resources* (3.1.32) that make up  
 502 the *Device* (3.1.13)

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

504 **3.1.22**  
 505 **Links**  
 506 extends typed web links

507 [SOURCE: IETF RFC 8288]

508 **3.1.23**  
 509 **Non-Discoverable Resource**  
 510 *Resource* (3.1.32) that is not listed in "/oic/res"

511 Note 1 to entry: The *Resource* (3.1.32) can be reached by a *Link* (3.1.22) which is conveyed by another *Resource*  
 512 (3.1.32). For example a *Resource* (3.1.32) linked in a *Collection* (3.1.7) does not have to be listed in "/oic/res", since  
 513 traversing the *Collection* (3.1.7) would discover the *Resource* (3.1.32) implemented on the *Device* (3.1.13).

514 **3.1.24**  
 515 **Notification**  
 516 mechanism to make a *Client* (3.1.6) aware of state changes in a *Resource* (3.1.32)

517 **3.1.25**  
 518 **Observe**  
 519 act of monitoring a *Resource* (3.1.32) by sending a RETRIEVE operation which is cached by the  
 520 *Server* (3.1.37) hosting the *Resource* (3.1.32) and reprocessed on every change to that *Resource*  
 521 (3.1.32)

522 **3.1.26**  
 523 **OpenAPI 2.0**  
 524 *Resource* (3.1.32) and *Introspection Device Data* (3.1.21) definitions used in this document

525 [SOURCE: OpenAPI specification]

526 **3.1.27**  
 527 **Parameter**  
 528 element that provides metadata about a *Resource* (3.1.32) referenced by the target URI of a *Link*  
 529 (3.1.22)

530 **3.1.28**  
 531 **Partial UPDATE**  
 532 UPDATE operation to a *Resource* (3.1.32) that includes a subset of the *Properties* (3.1.34) that are  
 533 visible via the *OCF Interface* (3.1.19) being applied for the *Resource Type* (3.1.35)



534 **3.1.29**  
535 **Permanent Immutable ID**  
536 identity for a *Device* (3.1.13) that cannot be altered

537 **3.1.30**  
538 **Physical Device**  
539 physical thing on which a *Device(s)* (3.1.13) is exposed

540 **3.1.31**  
541 **Platform**  
542 *Physical Device* (3.1.30) containing one or more *Devices* (3.1.13)

543 **3.1.32**  
544 **Resource**  
545 represents an entity modelled and exposed by the *Framework* (3.1.18)

546 **3.1.33**  
547 **Resource Interface**  
548 qualification of the permitted requests on a *Resource* (3.1.32)

549 **3.1.34**  
550 **Property**  
551 significant aspect or *Parameter* (3.1.27) of a *Resource* (3.1.32), including metadata, that is exposed  
552 through the *Resource* (3.1.32)

553 **3.1.35**  
554 **Resource Type**  
555 uniquely named definition of a class of *Properties* (3.1.34) and the interactions that are supported  
556 by that class

557 Note 1 to entry: Each *Resource* (3.1.32) has a *Property* (3.1.34) "rt" whose value is the unique name of the *Resource*  
558 *Type* (3.1.35).

559 **3.1.36**  
560 **Secure OCF Endpoint**  
561 *OCF Endpoint* (3.1.17) with a secure connection (e.g., CoAPS)

562 **3.1.37**  
563 **Semantic Tag**  
564 meta-information that provides additional contextual information with regard to the *Resource*  
565 (3.1.32) that is the target of a *Link* (3.1.22)

566 **3.1.38**  
567 **Server**  
568 *Device* (3.1.13) with the role of providing *Resource* (3.1.32) state information and facilitating remote  
569 interaction with its *Resources* (3.1.32)

570 **3.1.39**  
571 **Sleepy Server**  
572 *Server* (3.1.38) that will have latency in responding to requests

573 **3.1.40**  
574 **Unsecure OCF Endpoint**  
575 *OCF Endpoint* (3.1.17) with an unsecure connection (e.g., CoAP)

576 **3.1.41**  
577 **Vertical Resource Type**  
578 *Resource Type* (3.1.35) in a vertical domain specification

579 Note 1 to entry: An example of a Vertical *Resource Type* (3.1.41) would be "oic.r.switch.binary".

## 580 **3.2 Symbols and abbreviated terms**

581	ACL	Access Control List
582	BLE	Bluetooth Low Energy
583	CBOR	Concise Binary Object Representation
584	CoAP	Constrained Application Protocol
585	CoAPs	Secure Constrained Application Protocol
586	DTLS	Datagram Transport Layer Security
587	IP	Internet Protocol
588	ISP	Internet Service Provider
589	JSON	JavaScript Object Notation
590	MTU	Maximum Transmission Unit
591	OCF	Open Connectivity Foundation
592	REST	Representational State Transfer
593	RESTful	REST-compliant Web services
594	UDP	User Datagram Protocol
595	URI	Uniform Resource Identifier
596	UUID	Universal Unique Identifier

## 597 **4 Document conventions and organization**

### 598 **4.1 Conventions**

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

603 In this document, to be consistent with the IETF usages for RESTful operations, the RESTful  
604 operation words CRUDN, CREATE, RETRIVE, UPDATE, DELETE, and NOTIFY will have all letters  
605 capitalized. Any lowercase uses of these words have the normal technical English meaning.

606 The messaging payload examples in this document contain OCF Vertical Device Types and  
607 Resource Types, which are used for illustrative purposes only.

### 608 **4.2 Notation**

609 In this document, features are described as required, recommended, allowed or DEPRECATED as  
610 follows:

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

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

Recommended (or should)(S).

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

Allowed (may or allowed)(O).

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

DEPRECATED.

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

Conditionally allowed (CA).

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

Conditionally required (CR).

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

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

Words that are emphasized are printed in *italic*.

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

### 4.3 Data types

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

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

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

**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)?)?))\$ ^([0-9]+W)\$ ^([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[0-9]):([0-5] 0[0-9])\$ ^([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[0-9])([0-5] 0[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.

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

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

#### 4.4 Resource notation syntax

When it is desired to describe the Property of a Resource Type or the "anchor" Parameter value in an abbreviated notation, it can be described as follows:

- A value of the "rt" Property of the Resource Type or "anchor" Parameter value ":" Property name
- e.g., "oic.wk.d:di", which is the "di" Property of the Device Resource Type.

If Property name is a composite type (a type that is composed of several Properties), it can be described in recursive way. The following expression describes this as a regular expression format:

- A value of the "rt" Property of the Resource Type or "anchor" Parameter value (":" Property name )+
- e.g., "oic.r.pstat:dos:s", which is the "s" Property of the "dos" Property of the "pstat" Resource Type (see 13.8 of ISO/IEC 30118-2).

If there is a Resource URI (i.e., The Resource instance for a specific Resource Type), it can be used instead of using a value of "rt" Property of Resource Type or the "anchor" Parameter value as follows:

- A Resource URI (":" Property name )+
- e.g., "/oic/d:di", which is the "di" Property of the Device Resource Type instance.
- e.g., "/oic/sec/pstat:dos:s", which is the "s" Property of the "dos" Property of the "oic.r.pstat" Resource Type instance.

In the auto-generated Annex's Property definition tables for Resource Types, the Property names can be noted as belonging to the RETRIEVE schema or to the UPDATE schema by prefixing the Property name with "RETRIEVE" or "UPDATE" followed with the ":" separator. This is to avoid duplicate Property names appearing in the Property definition tables that are auto-generated. The following are examples using this notation with the "locn" Property of the "oic.wk.con" Resource Type:

- "RETRIEVE:locn"
- "UPDATE:locn"

## 5 Architecture

### 5.1 Overview

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

Specifically, the architecture provides:

- A communication and interoperability framework for multiple market segments (Consumer, Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication, transports and use cases.
- A common and consistent model for describing the environment and enabling information and semantic interoperability.
- Common communication protocols for discovery and connectivity.
- Common security and identification mechanisms.
- Opportunity for innovation and product differentiation.

- A scalable solution addressing different Device capabilities, applicable to smart devices as well as the smallest connected things and wearable devices.

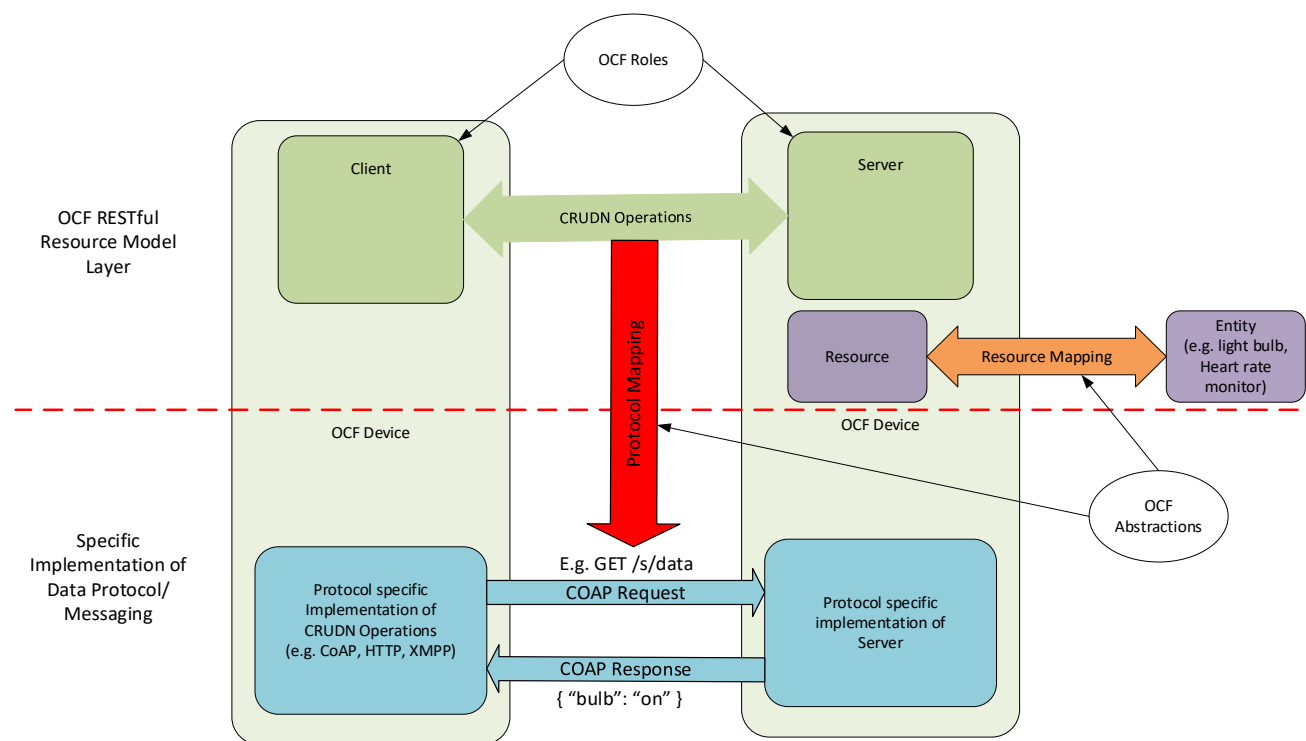
The architecture is based on the Resource Oriented Architecture design principles and described in the 5.2 through 5.4 respectively. 5.2 presents the guiding principles for OCF operations. 5.3 defines the functional block diagram and Framework.

## 5.2 Principle

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

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

Figure 1 depicts the architecture.



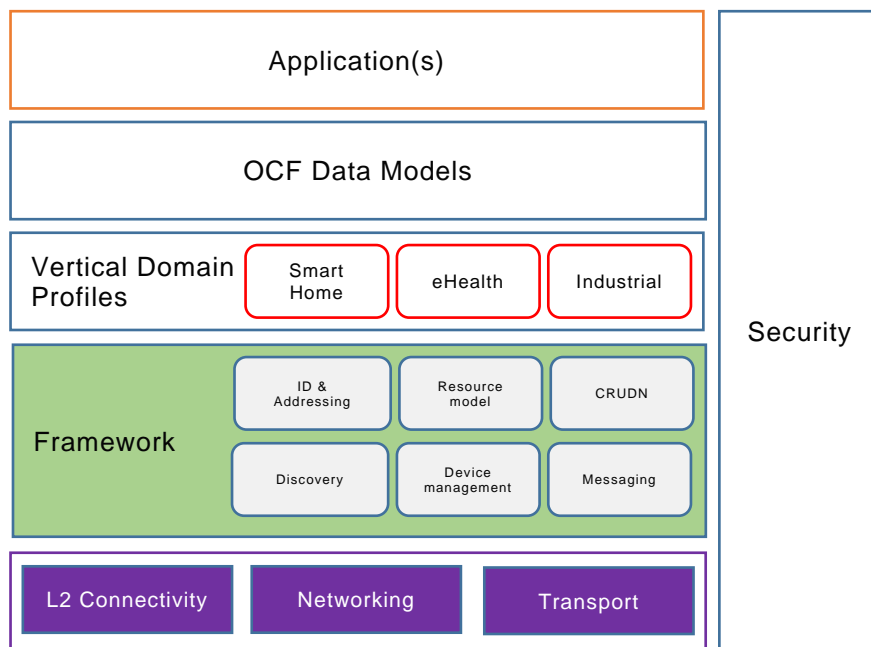
**Figure 1 – Architecture - concepts**

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

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

### 5.3 Functional block diagram

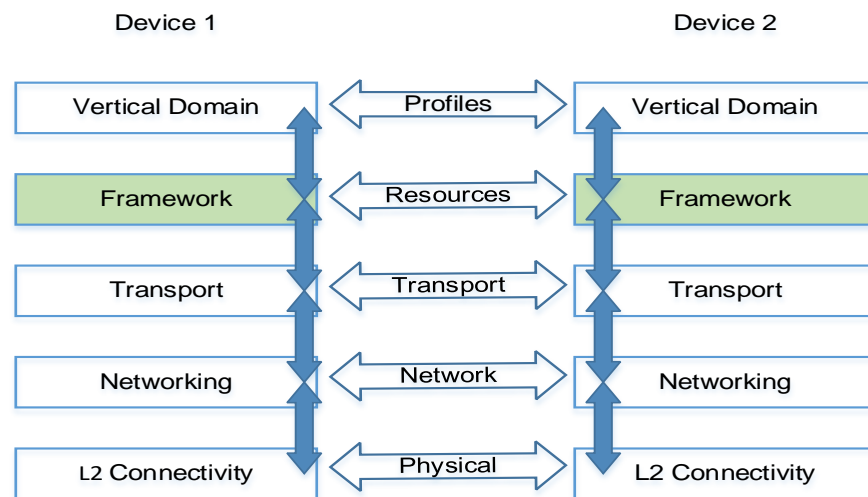
The functional block diagram encompasses all the functionalities required for operation. These functionalities are categorized as L2 connectivity, networking, transport, Framework, and application profiles. The functional blocks are depicted in Figure 2.



**Figure 2 – Functional block diagram**

- **L2 connectivity:** Provides the functionalities required for establishing physical and data link layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.
- **Networking:** Provides functionalities required for Devices to exchange data among themselves over the network (e.g., Internet).
- **Transport:** Provides end-to-end flow transport with specific QoS constraints. Examples of a transport protocol include TCP and UDP or new Transport protocols under development in the IETF, e.g., Delay Tolerant Networking (DTN).

- *Framework*: Provides the core functionalities as defined in this document. The functional block is the source of requests and responses that are the content of the communication between two Devices.
  - *Vertical Domain profile*: Provides market segment specific functionalities, e.g., functions for the smart home market segment.
- When two Devices communicate with each other, each functional block in a Device interacts with its counterpart in the peer Device as shown in Figure 3.



**Figure 3 – Communication layering model**

#### 5.4 Framework

Framework consists of functions which provide core functionalities for operation.

- *Identification and addressing*. Defines the identifier and addressing capability. The Identification and addressing function is defined in clause 6.
- *Discovery*. Defines the process for discovering available.
  - Devices (OCF Endpoint Discovery in clause 10) and
  - Resources (Resource discovery in 11.2).
- *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.
- *CRUDN*. Provides a generic scheme for the interactions between a Client and Server as defined in clause 8.
- *Messaging*. Provides specific message protocols for RESTful operation, i.e. CRUDN. For example, CoAP is a primary messaging protocol. The messaging function is defined in 11.5.
- *Security*. Includes authentication, authorization, and access control mechanisms required for secure access to Entities. The security function is defined in clause 13.



## 6 Identification and addressing

### 6.1 Introduction

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

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.

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.

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

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

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

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

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

### 6.2 Identification

#### 6.2.1 Device and Platform identification

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

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

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

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

#### 6.2.2 Resource identification and addressing

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

827 An OCF URI is based on the general form of a URI as defined in IETF RFC 3986 as follows (note  
828 that the portion in square brackets is optional):

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

830 Specifically, the OCF URI is specified in the following form:

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

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

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

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

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

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

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

852 The "query" is a string that shall contain one or more "<name>=<value>" constructs (aka name-  
853 value pair). Where multiple such constructs are supported, each is separated by an "&"  
854 (ampersand); this is not a logical "and" operation, but purely a delimiter. Where the use of a query  
855 is supported, how the query is handled by the recipient thereof is explicitly defined by the relevant  
856 clause in this document or other specifications. The query string will be mapped to the appropriate  
857 syntax of the protocol used for messaging. (e.g., CoAP).

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

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

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

### 6.3 Namespace:

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

### 6.4 Network addressing

The following are the addresses used in this document:

IP address

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

## 7 Resource model

### 7.1 Introduction

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

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

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

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

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

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

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

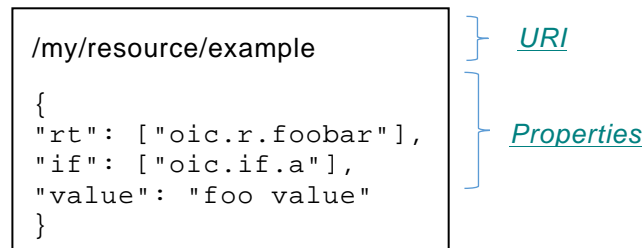
The information for Request or Response with the Representation may be communicated on the wire by serializing using a transfer protocol or encapsulated in the payload of the transport protocol – the specific method is determined by the normative mapping of the Request or Response to the transport protocol.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

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

## 7.2 Resource

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

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



**Figure 4 – Example Resource**

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

## 7.3 Property

### 7.3.1 Introduction

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

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

In addition, the Property definition shall have a

- *Value Type* – the Value Type defines the values that a Property Value may take. The Value Type may be a simple data type (e.g. string, Boolean) as defined in 4.3 or may be a complex data type defined with a schema. The Value Type may define
  - Value Rules define the rules for the set of values that the Property Value may take. Such rules may define the range of values, the min-max, formulas, the set of enumerated values, patterns, conditional values, and even dependencies on values of other Properties. The rules may be used to validate the specific values in a Property Value and flag errors.
- *Mandatory* – specifies if the Property is mandatory or not for a given Resource Type.

- *Access modes* – specifies whether the Property may be read, written or both. Updates are equivalent to a write. "r" is used for read and "w" is used for write – both may be specified. Write does not automatically imply read.

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

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

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

## **7.3.2 Common Properties**

### **7.3.2.1 Introduction**

The mandatory Common Properties defined in clause 7.3.2 shall be exposed and the optional Common Properties may be exposed in all Resources. The following Properties are defined as Common Properties:

The Common Properties for all Resources are specified in 7.3.2.3 through 7.3.2.6 respectively and summarized as follows:

- *Resource Type* ("rt") – this mandatory Property is used to declare the Resource Type of that Resource. Since a Resource could be defined by more than one Resource Type the Property Value of the Resource Type Property may be used to declare more than one Resource Type (see clause 7.4.4). See 7.3.2.3 for details.
- *OCF Interface* ("if") – this mandatory Property declares the OCF Interfaces supported by the Resource. The Property Value of the OCF Interface Property may be multi-valued and lists all the OCF Interfaces supported. See 7.3.2.4 for details.
- *Name* ("n") – this optional Property declares human-readable name assigned to the Resource. See 7.3.2.5.
- *Resource Identity* ("id") – this optional Property Value shall be a unique (across the scope of the host Server) identifier for a specific instance of the Resource. The encoding of this identifier is Device and implementation dependent. See 7.3.2.6 for details.

An optional Common Property may be mandatory when explicitly specified in a particular Resource Type definition (e.g., the "n" Common Property for the "oic.wk.d" Resource Type).

The name of a Common Property is unique and is not used by other Properties. When defining a new Resource Type, its non-common Properties will not use the name of existing Common Properties (e.g., "rt", "if", "n", and "id").

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

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

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

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

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

### 7.3.2.3 Resource Type

Resource Type Property is specified in 7.4.

### 7.3.2.4 OCF Interface

OCF Interface Property is specified in 7.6.

### 7.3.2.5 Name

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

**Table 2 – Name Property Definition**

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

Note: This Property may be mandatory when specifically defined for a Resource Type (e.g., "oic.wk.d").

The Name Property is read-write unless otherwise restricted by the Resource Type (i.e. the Resource Type does not support UPDATE or does not support UPDATE using the read-write OCF Interface ("oic.if.rw")).

### 7.3.2.6 Resource Identity

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

**Table 3 – Resource Identity Property Definition**

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

Note: This Property may be mandatory when specifically defined for a Resource Type.

## 7.4 Resource Type

### 7.4.1 Introduction

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

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

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

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

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

#### 7.4.2 Resource Type Property

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

**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

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

#### 7.4.3 Resource Type definition

Resource Type is specified as follows:

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

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

**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

**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

For example, an instance of the foobar Resource Type.

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

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

```
{
  "$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "rt": {
      "type": "array",
      "items": {
        "type": "string",
        "maxLength": 64
      },
      "minItems": 1,
      "readOnly": true,
      "description": "Resource Type of the Resource"
    },
    "if": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.lb", "oic.if.rw", "oic.if.r", "oic.if.a", "oic.if.s"]
      },
      "value": {"type": "string"}
    }
  }
}
```



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

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

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

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

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

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

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

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

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

#### 1130 **7.5 Device Type**

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

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

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

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

## 7.6 OCF Interface

### 7.6.1 Introduction

An OCF Interface provides first a view into the Resource and then defines the requests and responses permissible on that view of the Resource. So this view provided by an OCF Interface defines the context for requests and responses on a Resource. Therefore, the same request to a Resource when targeted to different OCF Interfaces may result in different responses. Depending on the view requested (i.e., OCF Interface), the Resource representation may not include all mandatory Properties (e.g., the "rt" and "if" Common Properties). If Common Properties are desired in the view requested, use the "oic.if.baseline" OCF Interface (see clause 7.6.3.2) which every Resource Type shall implement.

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

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

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

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

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

### 7.6.2 OCF Interface Property

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

**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.

### 7.6.3 OCF Interface methods

#### 7.6.3.1 Overview

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

The defined OCF Interfaces are listed in Table 8:

**Table 8 – OCF standard OCF Interfaces**

OCF Interface	Name	Applicable Operations	Description
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 Common 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.

#### 7.6.3.2 Baseline OCF Interface

##### 7.6.3.2.1 Overview

The Representation that is visible using the baseline OCF Interface includes all the Properties of the Resource including the mandatory and implemented optional Common Properties. The baseline OCF Interface shall be defined for 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.

#### 7.6.3.2.2 Use of RETRIEVE

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

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

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

#### 7.6.3.2.3 Use of UPDATE

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

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

### 7.6.3.3 Links list OCF Interface

#### 7.6.3.3.1 Overview

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

#### 7.6.3.3.2 Use with RETRIEVE

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

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

```
RETRIEVE /scenes/scenel?if=oic.if.ll
```

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

```
Response: Content
Payload:
[
  {
    "href": "/the/light/1",
    "rt": ["oic.r.switch.binary"],
    "if": ["oic.if.a", "oic.if.baseline"],

```

```

1242     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1243   },
1244   {
1245     "href": "/the/light/2",
1246     "rt": ["oic.r.switch.binary"],
1247     "if": ["oic.if.a", "oic.if.baseline"],
1248     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1249   },
1250   {
1251     "href": "/my/fan/1",
1252     "rt": ["oic.r.switch.binary"],
1253     "if": ["oic.if.a", "oic.if.baseline"],
1254     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1255   },
1256   {
1257     "href": "/his/fan/2",
1258     "rt": ["oic.r.switch.binary"],
1259     "if": ["oic.if.a", "oic.if.baseline"],
1260     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1261   }
1262 ]
1263

```

#### 7.6.3.3.3 Use with NOTIFY

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

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

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

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

```

1277 Response: Content
1278 Payload:
1279 [
1280   {
1281     "href": "/the/light/1",
1282     "rt": ["oic.r.switch.binary"],
1283     "if": ["oic.if.a", "oic.if.baseline"],
1284     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1285   },
1286   {
1287     "href": "/the/light/2",
1288     "rt": ["oic.r.switch.binary"],
1289     "if": ["oic.if.a", "oic.if.baseline"],
1290     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1291   },
1292   {
1293     "href": "/my/fan/1",
1294     "rt": ["oic.r.switch.binary"],
1295     "if": ["oic.if.a", "oic.if.baseline"],
1296     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1297   },
1298   {

```

```

1299     "href": "/his/fan/2",
1300     "rt": ["oic.r.switch.binary"],
1301     "if": ["oic.if.a", "oic.if.baseline"],
1302     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1303   }
1304 ]
1305

```

1306 Later when the "/his/fan/2" Link is removed (e.g., UPDATE operation with the Link remove OCF  
 1307 Interface) the response to the NOTIFY operation request is as illustrated:

```

1308 Response: Content
1309 Payload:
1310 [
1311   {
1312     "href": "/the/light/1",
1313     "rt": ["oic.r.switch.binary"],
1314     "if": ["oic.if.a", "oic.if.baseline"],
1315     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1316   },
1317   {
1318     "href": "/the/light/2",
1319     "rt": ["oic.r.switch.binary"],
1320     "if": ["oic.if.a", "oic.if.baseline"],
1321     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1322   },
1323   {
1324     "href": "/my/fan/1",
1325     "rt": ["oic.r.switch.binary"],
1326     "if": ["oic.if.a", "oic.if.baseline"],
1327     "eps": [{"ep": "coaps://[2001:db8:a::b1d4]:55555"}]
1328   }
1329 ]

```

1330 If the result of removing a Link results in no Links being present, then an empty array list shall be  
 1331 sent in a notification. An example of a response with no Links being present is as illustrated:

```

1332 Response: Content
1333 Payload:
1334 [
1335 ]

```

#### 1336 7.6.3.3.4 Use with CREATE, UPDATE, and DELETE

1337 The CREATE, UPDATE and DELETE operations are not allowed by the Links list OCF Interface.  
 1338 Attempts to perform CREATE, UPDATE or DELETE operations using the Links list OCF Interface  
 1339 shall return an appropriate error status code, for example "Method Not Allowed".

#### 1340 7.6.3.4 Batch OCF Interface

##### 1341 7.6.3.4.1 Overview

1342 The batch OCF Interface is used to interact with a Collection of Resources using a single/same  
 1343 Request. The batch OCF Interface can be used to RETRIEVE or UPDATE the Properties of the  
 1344 linked Resources with a single request.

##### 1345 7.6.3.4.2 General requirements for realizations of the batch OCF Interface

1346 All realisations of the batch OCF Interface adhere to the following:

- 1347 – The batch OCF Interface name is "oic.if.b"
- 1348 – A Collection Resource has linked Resources that are represented as URIs. In the "href"  
 1349 Property of the batch payload the URI shall be fully qualified for remote Resources and a  
 1350 relative reference for local Resources.

- 1351 – The original request is modified to create new requests targeting each of the linked Resources  
1352 in the Collection by substituting the URI in the original request with the URI of the linked  
1353 Resource. The payload in the original request is replicated in the payload of the new requests.
- 1354 – The requests shall be forwarded assuming use of the Default OCF Interface of the linked  
1355 Resources.
- 1356 – Requests shall only be forwarded to linked Resources that are identified by relation types "item"  
1357 or "hosts" ("hosts" is the default relation type value should the "rel" Link Parameter not be  
1358 present). Requests shall not be forwarded to linked Resources that do not contain the "item" or  
1359 "hosts" relation type values.
- 1360 – Properties of the Collection Resource itself may be included in payloads using "oic.if.b" OCF  
1361 Interface by exposing a single Link with the link relation "self" along with "item" within the  
1362 Collection, and ensuring that Link resolution cannot become an infinite loop due to recursive  
1363 references. For example, if the Default OCF Interface of the Collection is "oic.if.b", then the  
1364 Server might recursively include its batch representation within its batch representation, in an  
1365 endless loop. See 7.6.3.4.5 for an example of use of a Link containing "rel": ["self", "item"] to  
1366 include Properties of the Collection Resource, along with linked Resources, in "oic.if.b"  
1367 payloads.
- 1368 – If the Default OCF Interface of a Collection Resource is exposed using the Link relation "self",  
1369 and the Default OCF Interface contains Properties that expose any Links, those Properties shall  
1370 not be included in a batch representation which includes the "self" Link.
- 1371 – Any request forwarded to a linked Resource that is a Collection (including a "self" Link reference)  
1372 shall have the Default OCF Interface of the linked Collection Resource applied.
- 1373 – All the responses from the linked Resources shall be aggregated into a single Response to the  
1374 Client. The Server may timeout the response to a time window, the Server may choose any  
1375 appropriate window based on conditions.
- 1376 – If a linked Resource cannot process the request, an empty response, i.e. a JSON object with  
1377 no content ("{}") as the representation for the "rep" Property, or error response should the linked  
1378 Resource Type provide an error schema or diagnostic payload, shall be returned by the linked  
1379 Resource. These empty or error responses for all linked Resources that exhibit an error shall  
1380 be included in the aggregated response to the original Client request. See the example in  
1381 7.6.3.4.5.
- 1382 – If any of the linked Resources returns an error response, the aggregated response sent to the  
1383 Client shall also indicate an error (e.g. 4.xx in CoAP). If all of the linked Resources return  
1384 successful responses, the aggregated response shall include the success response code.
- 1385 – The aggregated response shall be an array of objects representing the responses from each  
1386 linked Resource. Each object in the response shall include at least two items: (1) the URI of  
1387 the linked Resource (fully qualified for remote Resources, or a relative reference for local  
1388 Resources) as "href": <URI> and (2) the individual response object or array of objects if the  
1389 linked Resource is itself a Collection using "rep" as the key, e.g. "rep": { <representation of  
1390 individual response> }.
- 1391 – The Client may specify the Resource Type(s) of the linked Resources to which the request is  
1392 forwarded by including one or more "rt" query parameters in the request, each separated by an  
1393 "&" as a delimiter (e.g. "?if=oic.if.b&rt=oic.r.switch.binary"). The Server shall then process such  
1394 additional query parameters in a request that includes "oic.if.b", as selectors for the Linked  
1395 Resources that are to be processed by the request.

#### 1396 **7.6.3.4.3 Observability of the batch OCF Interface**

1397 When a Collection supports the ability to be observed using the batch OCF Interface the following  
1398 apply:

- 1399 – If the Collection Resource is marked as Observable, linked Resources referenced in the  
1400 Collection may be Observed using the batch OCF Interface. If the Collection Resource is not

marked as Observable then the Collection cannot be Observed and Observe requests to the Collection shall be handled as defined for the case where request validation fails in clause 11.3.2.4. The Observe mechanism shall work as defined in 11.3.2 with the Observe request forwarded to each of the linked Resources. All responses to the request shall be aggregated into a single response to the Client using the same representations and status codes as for RETRIEVE operations using the batch OCF Interface.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- The UPDATE response shall contain the updated values using the same payload schema as RETRIEVE operations if provided by the linked Resource, along with the appropriate status



1450 code. The aggregated response payload shall reflect the known state of the updated Properties  
1451 after the batch update was completed. If no payload is provided by the updated Resource, then  
1452 an empty response (i.e. "rep": {}) shall be provided for that Resource.

1453 – A Collection shall not support the use of the UPDATE operation to add, modify, or remove Links  
1454 in an existing Collection using the "oic.if.baseline", "oic.if.rw" or "oic.if.a" OCF Interfaces.

1455 – A Collection shall not support the use of the UPDATE operation using the batch OCF Interface  
1456 when the Collection contains Links that resolve to Resources that are not hosted on the Device  
1457 that also hosts the Collection. If such a Collection receives an UPDATE operation, the operation  
1458 shall be rejected with a response indicating an appropriate error (e.g. method not allowed). If  
1459 the ability to UPDATE linked remote Resources is desired, the use of the optional scene feature  
1460 (see clause 11.6 in [1]) to effect the UPDATE could be utilized.

#### 1461 **7.6.3.4.5 Examples: Batch OCF Interface**

1462 Note that the examples provided in Table 9 are illustrative and do not include all mandatory schema  
1463 elements in all cases. It is assumed that the Default OCF Interface for the Resource Type  
1464 "x.org.example.rt.room" is specified in its Resource Type definition file as "oic.if.rw", which exposes  
1465 the Properties "x.org.example.colour" and "x.org.example.size".

**Table 9 – Batch OCF Interface Example**

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

	<pre> /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}     }     {       "href": "/switch-1b",       "rt": ["oic.r.switch.binary"],       "if": ["oic.if.a", "oic.if.baseline"],       "p": {"bm": 2 }     }   ] } </pre>
--	--

<b>Use of batch, successful response</b>	<p>Request: GET /a/room/1?if=oic.if.b</p> <p>Becomes the following individual request messages issued by the Device in the Client role</p> <p>GET /a/room/1 (NOTE: uses the Default OCF Interface as specified for the Collection Resource, in this example oic.if.rw)</p> <p>GET /the/light/1 (NOTE: Uses the Default OCF Interface as specified for this Resource)</p> <p>GET /the/light/2 (NOTE: Uses the Default OCF Interface as specified for this Resource)</p> <p>GET /my/fan/1 (NOTE: Uses the Default OCF Interface as specified for this Resource)</p> <p>GET /his/fan/2 (NOTE: Uses the Default OCF Interface as specified for this Resource)</p> <p>GET /the/presence/1 (NOTE: Uses the Default OCF Interface as specified for this Resource)</p> <p>GET /the/switches/1 (NOTE: Uses the Default OCF Interface for the Collection that is within the Collection)</p> <p>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": { "value": true }   },   {     "href": "/his/fan/2",     "rep": { "value": false }   },   {     "href": "/the/presence/1",     "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]:55555" }         ]       },       {         "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]:55555" }         ]       }     ]   } ]</pre>
--	---

	<div>]</div> <div>}</div> <div>}</div> <div>}</div> <div>}</div>
--	--

<b>Use of batch, error response</b>	<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/presence/1",     "rep": {"value": false}   },   {     "href": "/the/switches/1",     "rep": {}   } ]</pre>
-------------------------------------	--

<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>Retrieving all items that are Presence Sensors ("oic.r.sensor.presence"):  RETRIEVE /a/room/1?if=oic.if.b&amp;rt=oic.r.sensor.presence</p> <p>Response payload:</p> <pre> [   {     "href": "/the/presence/1",     "rep": {       "value": false     }   } ] </pre>
--	--



### 7.6.3.5 Actuator OCF Interface

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

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

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

```
/a/act/heater
{
  "rt": ["x.com.acme.gas"],
  "if": ["oic.if.baseline", "oic.if.r", "oic.if.a", "oic.if.s"],
  "x.com.acme.settemp": 10,
  "x.com.acme.currenttemp" : 7
}
```

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

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

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

Response: Content

Payload:

```
{
  "x.com.acme.settemp": 10,
  "x.com.acme.currenttemp" : 7
}
```

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

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

```
{
  "x.com.acme.settemp": 20
}
```

Response: Changed

Payload:

```
{
  "x.com.acme.settemp": 20
}
```

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

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

```
{
  "if": ["oic.if.s"]    ← this is visible through baseline OCF Interface
}
```

Response: Bad Request

Payload:

```
{
}
```

- A RETRIEVE request using this OCF Interface shall return the Representation for this Resource as defined by the applicable OpenAPI 2.0 schema, subject to any query parameters that may also be defined as part of the applicable OpenAPI 2.0 schema.
- An UPDATE request using this OCF Interface shall provide a payload or body that contains the Properties that will be updated on the target Resource.

### 7.6.3.6 Sensor OCF Interface

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

- The sensor OCF Interface name shall be "oic.if.s".
- The sensor OCF Interface shall expose in the Resource Representation all mandatory Properties as defined by the applicable OpenAPI 2.0 schema; the sensor OCF Interface may also expose in the Resource Representation optional Properties as defined by the applicable OpenAPI 2.0 schema that are implemented by the target Device.
- A RETRIEVE request using this OCF Interface shall return this representation for the Resource as defined by the applicable OpenAPI 2.0 schema, subject to any query parameters that may also be defined as part of the applicable OpenAPI 2.0 schema.

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

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

Response: Content

Payload:

```
{
  "x.com.acme.currenttemp": 7
}
```

Incorrect use of the sensor.

Request: UPDATE /a/act/heater?if="oic.if.s" ← UPDATE is not allowed

```
{
  "x.com.acme.settemp": 20 ← this is possible through actuator OCF Interface
}
```

Response: Bad Request

Payload:

```
{
}
```

Another incorrect use of the sensor.

Request: UPDATE /a/act/heater?if="oic.if.s" ← UPDATE is not allowed

```
{
  "x.com.acme.currenttemp": 15 ← this is not possible to be updated
}
```

Response: Bad Request

Payload:

```
{
}
```

### 7.6.3.7 Read-only OCF Interface

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

The read-only OCF Interface with respect to "Heater" Resource (for illustration only):

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

Response: Content

Payload:

```
{
```

```
1572     "x.com.acme.settemp": 10,  
1573     "x.com.acme.currenttemp" : 7  
1574 }
```

### 1575 **7.6.3.8 Read-write OCF Interface**

1576 The read-write OCF Interface is a generic OCF Interface to support reading and setting Properties  
1577 in a Resource. The applicable methods that can be applied to a Resource are only RETRIEVE,  
1578 NOTIFY, and UPDATE. For the RETRIEVE and NOTIFY operations, the behaviour is the same as  
1579 for the "oic.if.r" OCF Interface defined in 7.6.3.7. For the UPDATE operation, read-only Properties  
1580 (i.e. Properties tagged with "readOnly=true" in the OpenAPI 2.0 definition) shall not be in the  
1581 UPDATE payload. An attempt by a Client to apply a method other than RETRIEVE, NOTIFY, or  
1582 UPDATE to a Resource shall be rejected with an error response code.

1583 For example, a "Grinder" Resource (for illustration only):

```
1584 /a/mygrinder  
1585 {  
1586     "rt": ["oic.r.grinder"],  
1587     "if": ["oic.if.rw", "oic.if.baseline"],  
1588     "coarseness": 10,  
1589     "remaining": 50  
1590 }
```

1591

1592 The read-write OCF Interface with respect to "Grinder" Resource (for illustration only):

#### 1593 **a) Retrieving the value with read-write OCF Interface**

```
1594  
1595 Request: RETRIEVE /a/mygrinder?if="oic.if.rw"  
1596  
1597 Response: Content  
1598 Payload:  
1599 {  
1600     "coarseness": 10,  
1601     "remaining": 50  
1602 }  
1603
```

#### 1604 **b) Updating the value with read-write OCF Interface**

```
1605  
1606 Request: UPDATE /a/mygrinder?if="oic.if.rw"  
1607 {  
1608     "coarseness": 20  
1609 }  
1610  
1611 Response: Changed  
1612 Payload:  
1613 {  
1614     "coarseness": 20  
1615 }
```

### 1616 **7.6.3.9 Create OCF Interface**

#### 1617 **7.6.3.9.1 Overview**

1618 The create OCF Interface is used to create Resource instances in a Collection. An instance of a  
1619 Resource and the Link pointing to the Resource are created together, atomically, according to a  
1620 Client-supplied representation. The create OCF Interface name is "oic.if.create". A Collection which  
1621 exposes the "oic.if.create" OCF Interface shall expose the "rts" Property (see clause 7.8.2.8) with  
1622 all Resource Types that can be hosted with the Collection. If a Client attempts to create a Resource  
1623 Type which is not supported by the Collection, the Server shall return an appropriate error status

code, for example "Bad Request". Successful CREATE operations shall return a success code, i.e. "Created". The IDD for all allowed Resource Types that may be created shall adhere to Introspection for dynamic Resources (see clause 11.4).

#### 7.6.3.9.2 Data format for CREATE

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

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

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

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

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

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

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

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

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

```
{
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "p": {"bm": 3},
  "rep": {
    "temperature": 20
  }
}
```

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

```
{
  "href": "/3755f3ac",
  "rt": ["oic.r.temperature"],
  "if": ["oic.if.a", "oic.if.baseline"],
  "ins": 39724818,
  "p": {"bm": 3},
}
```

```

1671     "rep": {
1672         "rt": ["oic.r.temperature"],
1673         "if": ["oic.if.a", "oic.if.baseline"],
1674         "temperature": 20
1675     }
1676 }

```

1677 The Link Parameters representation in the "/oic/res" Resource, if the created Resource is  
 1678 discoverable, may not mirror exactly all the Link Parameters added in the Collection; except it shall  
 1679 expose at a minimum the mandatory Properties of the Link (i.e., "rt", "if", and "href") of the created  
 1680 Resource.

#### 1681 **7.6.3.9.3 Use with CREATE**

1682 The CREATE operation shall be sent to the URI of the Collection in which the Resource is to be  
 1683 created. The query string "?if=oic.if.create" shall be included in all CREATE operations.

1684 The Server shall generate a URI for the created Resource and include the URI in the "href"  
 1685 Parameter of the created Link.

1686 When a Server successfully completes a CREATE operation using the "oic.if.create" OCF Interface  
 1687 addressing a Collection, the Server shall automatically modify the ACL Resource to provide initial  
 1688 authorizations for accessing for the newly created Resource according to ISO/IEC 30118-2.

1689 An example performing a CREATE operation is as illustrated:

```

1690 CREATE /scenes/scenel?if=oic.if.create
1691 {
1692     "rt": ["oic.r.temperature"],
1693     "if": ["oic.if.a", "oic.if.baseline"],
1694     "p": {"bm": 3},
1695     "rep": {
1696         "temperature": 20
1697     }
1698 }
1699 Response: Created
1700 Payload:
1701 {
1702     "href": "/3755f3ac",
1703     "ins": 39724818,
1704     "rt": ["oic.r.temperature"],
1705     "if": ["oic.if.a", "oic.if.baseline"],
1706     "p": {"bm": 3},
1707     "rep": {
1708         "rt": ["oic.r.temperature"],
1709         "if": ["oic.if.a", "oic.if.baseline"],
1710         "temperature": 20
1711     }
1712 }

```

#### 1713 **7.6.3.9.4 Use with UPDATE and DELETE**

1714 The UPDATE and DELETE operations are not allowed by the create OCF Interface. Attempts to  
 1715 perform UPDATE or DELETE operations using the create OCF Interface shall return an appropriate  
 1716 error status code, for example "Method Not Allowed", unless the UPDATE and CREATE operations  
 1717 map to the same transport binding method (e.g., CoAP with the POST method). In that situation  
 1718 where the UPDATE and CREATE operations map to the same transport binding method, this shall  
 1719 be processed as a CREATE operation according to clause 7.6.3.9.3.

## 7.7 Resource representation

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

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

## 7.8 Structure

### 7.8.1 Introduction

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

### 7.8.2 Resource relationships (Links)

#### 7.8.2.1 Introduction

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

- Link context (URI reference) as defined in 7.8.2.2
- Link relation type as defined in 7.8.2.3
- Link target (URI reference) as defined in 7.8.2.4
- Link target attributes as defined in 7.8.2.5

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

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

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

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

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

Another example of a Link is as illustrated:

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

### 7.8.2.2 Link context

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

### 7.8.2.3 Link relation type

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

- "hosts"
  - The Link target points to a Resource that is hosted at the Link context. This Link relation type indicates that the Resource is allowed to be included in the batch representations of the Link target. This Link relation type is defined by IETF RFC 6690.
- "self"
  - The Link refers to the Link context, which allows a Link to describe the Resource at the Link context, which is to say that the Link can describe the Collection or Atomic Measurement Resource that the Link is retrieved from. The Link target points to the Link context, and the Link target attributes describe the Link context. This Link relation type is defined by IETF RFC 4287.
- "item"
  - The Link target points to a Resource that is a member of the Collection or Atomic Measurement at the Link context, which might not specifically be hosted by the Collection or Atomic Measurement Resource, and is allowed to be contained in batch representations of the Collection or Atomic Measurement. An example is using "rel": "item" to declare that the Properties of the Collection or Atomic Measurement Resource itself should be included in a batch representation of the Collection or Atomic Measurement. This Link relation type is defined by IETF RFC 6573.

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

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

### 7.8.2.4 Link target

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

### 7.8.2.5 Parameters for Link target attributes

#### 7.8.2.5.1 Introduction

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

**Table 10 – Link target attributes list**

Parameter title	Parameter name	Mandatory	Description
Device UUID	"di"	No	Defined in clause 7.8.2.5.5
OCF Endpoint information	"eps"	No	Defined in clause 7.8.2.5.6
OCF Interface	"if"	Yes	Defined in clause 7.6
Link instance	"ins"	No	Defined in clause 7.8.2.5.2
Policy	"p"	No	Defined in clause 7.8.2.5.3
Resource Type	"rt"	Yes	Defined in clause 7.4
Media type	"type"	No	Defined in clause 7.8.2.5.4
Position description Semantic Tag	"tag-pos-desc"	No	Defined in clause 11.5.2.1.2
Relative position Semantic Tag	"tag-pos-rel"	No	Defined in clause 11.5.2.1.3
Function description Semantic Tag	"tag-func-desc"	No	Defined in clause 11.5.2.2.2
Location description Semantic Tag	"tag-locn"	No	Defined in clause 11.5.2.3.2

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

#### 7.8.2.5.2 "ins" or Link instance Parameter

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

#### 7.8.2.5.3 "p" or policy Parameter

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

The policy Parameter "p" is defined by:

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

**Table 11 – "bm" Property definition**

Bit Position	Policy rule	Comment
Bit 0 (the LSB)	discoverable	<p>The discoverable rule defines whether the Link is to be included in the Resource discovery message via "/oic/res".</p> <p>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.</p> <p>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.</p>



Bit 1 (2 <sup>nd</sup> LSB)	observable	<p>The Observable rule defines whether the Resource referenced by the target URI supports the NOTIFY operation. With the self-link, i.e. the Link with "rel" value of "self", "/oic/res" can have a Link with the target URI of "/oic/res" and indicate itself Observable. The "self" is defined by IETF RFC 4287 and registered in the IANA Registry for "rel" value defined at IANA Link Relations.</p> <p>If the Resource supports the NOTIFY operation, then "p" shall include the "bm" key and set the Observable bit to value 1.</p> <p>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.</p>
Bits 2-7	--	Reserved for future use. All reserved bits in "bm" shall be set to value 0.

1821

1822 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  
1823 measure. However, if any bit is set to value 1, then "bm" shall be included in "p" and all the bits shall be defined  
1824 appropriately.

- 1825 – In a payload sent in response to a request that includes an OCF-Accept-Content-Format-  
1826 Version option the "eps" Parameter shall provide the information for an encrypted connection.
- 1827 – Note that access to the Resource is controlled by the ACL for the Resource. A successful  
1828 encrypted connection does not ensure that the requested action will succeed. See  
1829 ISO/IEC 30118-2 clause 12 for more information.

1830 This shows the policy Parameter for a Resource that is discoverable but not Observable.

1831 "p": { "bm": 1 }

1832 This shows a self-link, i.e. the "/oic/res" Link in itself that is discoverable and Observable.

```

1833 {
1834   "href": "/oic/res",
1835   "rel": "self",
1836   "rt": [ "oic.wk.res" ],
1837   "if": [ "oic.if.ll", "oic.if.baseline" ],
1838   "p": { "bm": 3 }
1839 }
```

#### 1840 7.8.2.5.4 "type" or media type Parameter

1841 The "type" Parameter may be used to specify the various media types that are supported by a  
1842 specific target Resource. The default type of "application/vnd.ocf+cbor" shall be used when the  
1843 "type" element is omitted. Once a Client discovers this information for each Resource, it may use  
1844 one of the available representations in the appropriate header field of the Request or Response.

#### 1845 7.8.2.5.5 "di" or Device UUID Parameter

1846 The "di" Parameter specifies the Device UUID of the Device that hosts the target Resource defined  
1847 in the in the "href" Parameter.

1848 The Device UUID may be used to qualify a relative reference used in the "href" or to lookup OCF  
1849 Endpoint information for the relative reference.

#### 1850 7.8.2.5.6 "eps" Parameter

1851 The "eps" Parameter indicates the OCF Endpoint information of the target Resource.

1852 A Device shall populate all exposed "eps" Link Parameters with an array of items representing OCF  
1853 Endpoint information as specified in 10.2. Each entry in that array shall include an "ep" Property,  
1854 and may include the optional "pri" and "lat" Properties.

1855 This is an example of "eps" with multiple OCF Endpoints.

```
1856 "eps": [  
1857   { "ep": "coap://[fe80::b1d6]:1111", "pri": 2, "lat": 240 },  
1858   { "ep": "coaps://[fe80::b1d6]:1122", "lat": 240 },  
1859   { "ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3 }  
1860 ]
```

1861 When "eps" is present in a link, the OCF Endpoint information in "eps" can be used to access the  
1862 target Resource referred by the "href" Parameter.

1863 Note that the type of OCF Endpoint – Secure or Unsecure – that a Resource exposes merely  
1864 determines the connection type(s) guaranteed to be available for sending requests to the Resource.  
1865 For example, if a Resource only exposes a single CoAP "ep", it does not guarantee that the  
1866 Resource cannot also be accessed via a Secure OCF Endpoint (e.g. via a CoAPS "ep" from another  
1867 Resource's "eps information). Nor does exposing a given type of OCF Endpoint ensure that access  
1868 to the Resource will be granted using the "ep" information. Whether requests to the Resource are  
1869 granted or denied by the Access Control layer is separate from the "eps" information, and is  
1870 determined by the configuration of the /acl2 Resource (see ISO/IEC 30118-2 clause 13.5.3 for  
1871 details).

1872 When present, max-age information (e.g. Max-Age option for CoAP defined in IETF RFC 7252)  
1873 determines the maximum time "eps" values may be cached before they are considered stale.

#### 1874 **7.8.2.6 Formatting**

1875 When formatting in JSON, the list of Links shall be an array.

#### 1876 **7.8.2.7 List of Links in a Collection**

1877 A Resource that exposes one or more Properties that are defined to be an array of Links where  
1878 each Link can be discretely accessed is a Collection. The Property Name "links" is recommended  
1879 for such an array of Links.

1880 This is an example of a Resource with a list of Links.

```
1881 /Room1  
1882 {  
1883   "rt": ["oic.wk.col"],  
1884   "if": ["oic.if.ll", "oic.if.baseline" ],  
1885   "color": "blue",  
1886   "links":  
1887   [  
1888     {  
1889       "href": "/switch",  
1890       "rt": ["oic.r.switch.binary"],  
1891       "if": [ "oic.if.a", "oic.if.baseline" ],  
1892       "p": { "bm": 3 }  
1893     },  
1894     {  
1895       "href": "/brightness",  
1896       "rt": ["oic.r.light.brightness"],  
1897       "if": [ "oic.if.a", "oic.if.baseline" ],  
1898       "p": { "bm": 3 }  
1899     }  
1900   ]  
1901 }
```

#### 1902 **7.8.2.8 Properties describing an array of Links**

1903 If a Resource Type that defines an array of Links (e.g. Collections, Atomic Measurements) has  
1904 restrictions on the "rt" values that can be within the array of Links, the Resource Type will define  
1905 the "rts" Property. The "rts" Property as defined in Table 12 will include all "rt" values allowed for

all Links in the array. If the Resource Type does not define the "rts" Property or the "rts" Property is an empty array, then any "rt" value is permitted in the array of Links.

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

**Table 12 – Resource Types Property definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
<b>Resource Types</b>	"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.

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

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

**Table 13 – 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.

### **7.8.3 Collections**

#### **7.8.3.1 Overview**

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

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

1938 of a Resource. Additional Resource Types and OCF Interfaces may be bound to the Collection at  
1939 creation or later during the lifecycle of the Collection.

1940 A Collection shall define a Property that is an array with zero or more Links. The target URIs in the  
1941 Links may reference another Collection or another Resource. The referenced Collection or  
1942 Resource may reside on the same Device as the Collection that includes that Link (called a local  
1943 reference) or may reside on another Device (called a remote reference). The context URI of the  
1944 Links in the array shall (implicitly) be the Collection that contains that Property. The (implicit)  
1945 context URI may be overridden with explicit specification of the "anchor" Parameter in the Link  
1946 where the value of "anchor" is the new base of the Link.

1947 A Resource may be referenced in more than one Collection, therefore, a unique parent-child  
1948 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the  
1949 Resource referenced in the Collection, i.e., the application may use Collections to represent a  
1950 relationship but none is automatically implied or defined. The lifecycles of the Collection and the  
1951 referenced Resource are also independent of one another.

1952 In the following example a Property "links" represents the list of Links in a Collection. The "links"  
1953 Property has, as its value, an array of items and each item is a Link.

```
1954 /my/house    ← This is URI of the Resource
1955 {
1956   "rt": ["my.r.house"],    ← This and the next 3 lines are the Properties of the
1957   Resource.
1958   "color": "blue",
1959   "n": "myhouse",
1960   "links": [
1961     {    ← This and the next 4 lines are the Parameters of a Link
1962       "href": "/door",
1963       "rt": ["oic.r.door"],
1964       "if": ["oic.if.a", "oic.if.baseline"]
1965     },
1966
1967     {
1968       "href": "/door/lock.status",
1969       "rt": ["oic.r.lock"],
1970       "if": ["oic.if.a", "oic.if.baseline"]
1971     },
1972
1973     {
1974       "href": "/light",
1975       "rt": ["oic.r.light"],
1976       "if": ["oic.if.s", "oic.if.baseline"]
1977     },
1978
1979     {
1980       "href": "/binarySwitch",
1981       "rt": ["oic.r.switch.binary"],
1982       "if": ["oic.if.a", "oic.if.baseline"]
1983     }
1984   ]
1985 }
1986
```

1987 A Collection may be:

- 1988 – A pre-defined Collection where the Collection has been defined a priori and the Collection is  
1989 static over its lifetime. Such Collections may be used to model, for example, an appliance that  
1990 is composed of other Devices or fixed set of Resources representing fixed functions.

- 1991 – A Device local Collection where the Collection is used only on the Device that hosts the  
1992 Collection. Such Collections may be used as a short-hand on a Client for referring to many  
1993 Servers as one.
- 1994 – A centralized Collection where the Collection is hosted on a Device but other Devices may  
1995 access or update the Collection.
- 1996 – A hosted Collection where the Collection is centralized but is managed by an authorized agent  
1997 or party.

### 1998 7.8.3.2 Collection Properties

1999 A Collection shall define a Property that is an array of Links (the Property Name "links" is  
2000 recommended). In addition, other Properties may be defined for the Collection by the Resource  
2001 Type. The mandatory and recommended Common Properties for a Collection are shown in Table 14.  
2002 This list of Common Properties is in addition to those defined for Resources in 7.3.2.

2003 **Table 14 – Common Properties for Collections (in addition to Common Properties defined**  
2004 **in 7.3.2)**

Property	Description	Property Name	Value Type	Mandatory
<b>Links</b>	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 12	As defined in Table 12	No
<b>Mandatory Resource Types</b>	The list of Resource Types for Links that are mandatory in the Collection.	As defined in Table 13	As defined in Table 13	No

2005

### 2006 7.8.3.3 Default Resource Type

2007 A default Resource Type, "oic.wk.col", is available for Collections. This Resource Type shall be  
2008 used only when another type has not been defined on the Collection or when no Resource Type  
2009 has been specified at the creation of the Collection.

2010 The default Resource Type provides support for the Common Properties including an array of Links  
2011 with the Property Name "links".

### 2012 7.8.3.4 Default OCF Interface

2013 All instances of a Collection shall support the links list ("oic.if.ll") OCF Interface in addition to the  
2014 baseline ("oic.if.baseline") OCF Interface. An instance of a Collection may optionally support  
2015 additional OCF Interfaces that are defined within this document. The Default OCF Interface for a  
2016 Collection shall be links list ("oic.if.ll") unless otherwise specified by the Resource Type definition.

## 2017 7.8.4 Atomic Measurement

### 2018 7.8.4.1 Overview

2019 Certain use cases require that the Properties of multiple Resources are only accessible as a group  
2020 and individual access to those Properties of each Resource by a Client is prohibited. The Atomic

Measurement Resource Type is defined to meet this requirement. This is accomplished through the use of the Batch OCF Interface.

#### 7.8.4.2 Atomic Measurement Properties

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

**Table 15 – Common Properties for Atomic Measurement (in addition to Common Properties 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 string then any Resource Type is permitted	As defined in Table 12	As defined in Table 12	No
<b>Mandatory Resource Types</b>	The list of Resource Types for Links that are mandatory in the Atomic Measurement.	As defined in Table 13	As defined in Table 13	No

#### 7.8.4.3 Normative behaviour

The normative behaviour of an Atomic Measurement is as follows:

- The behaviour of the Batch OCF Interface ("oic.if.b") on the Atomic Measurement is defined as follows:
  - Only RETRIEVE and NOTIFY operations are supported, for Batch OCF Interface, on Atomic Measurement; the behavior of the RETRIEVE and NOTIFY operations shall be the same as specified in 7.6.3.4, with exceptions as provided for in 7.8.4.3.
  - The UPDATE operation is not allowed, for Batch OCF Interface, on Atomic Measurement; if an UPDATE operation is received, it shall result in a method not allowed error code.
  - An error response shall not include any representation of a linked Resource (i.e. empty response for all linked Resources).
- Any linked Resource within an Atomic Measurement (i.e. the target Resource of a Link in an Atomic Measurement) is subject to the following conditions:
  - Linked Resources within an Atomic Measurement and the Atomic Measurement itself shall exist on a single Server.
  - CRUDN operations shall not be allowed on linked Resources and shall result in a forbidden error code.
  - Linked Resources shall not expose the "oic.if.ll" OCF Interface. Since CRUDN operations are not allowed on linked Resources, the "oic.if.ll" OCF Interface would never be accessible.

- 2051 – Links to linked Resources in an Atomic Measurement shall only be accessible through the  
2052 "oic.if.ll" or the "oic.if.baseline" OCF Interfaces of an Atomic Measurement.
- 2053 – The linked Resources shall not be listed in "/oic/res".
- 2054 – A linked Resource in an Atomic Measurement shall have defined one of "oic.if.a", "oic.if.s",  
2055 "oic.if.r", or "oic.if.rw" as its Default OCF Interface.
- 2056 – Not all linked Resources in an Atomic Measurement are required to be Observable. If an Atomic  
2057 Measurement is being Observed using the "oic.if.b" OCF Interface, notification responses shall  
2058 not be generated when the linked Resources which are not marked Observable are updated or  
2059 change state.
- 2060 – All linked Resources in an Atomic Measurement shall be included in every RETRIEVE and  
2061 Observe response when using the "oic.if.b" OCF Interface.
- 2062 – An Atomic Measurement shall support the "oic.if.b" and the "oic.if.ll" OCF Interfaces.
- 2063 – Filtering of linked Resources in an Atomic Measurement is not allowed. Query parameters that  
2064 select one or more individual linked Resources in a request to an Atomic Measurement shall  
2065 result in a "forbidden" error code.
- 2066 – If the "rel" Link Parameter is included in a Link contained in an Atomic Measurement, it shall  
2067 have either the "hosts" or the "item" value.
- 2068 – The Default OCF Interface of an Atomic Measurement is "oic.if.b".

#### 2069 7.8.4.4 Security considerations

2070 Access rights to an Atomic Measurement Resource Type is as specified in clause 12.2.7.2 (ACL  
2071 considerations for batch request to the Atomic Measurement Resource Type) of ISO/IEC 30118-2).

#### 2072 7.8.4.5 Default Resource Type

2073 The Resource Type is defined as "oic.wk.atomicmeasurement" as defined in Table 16.

2074 **Table 16 – Atomic Measurement Resource Type**

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

2075

2076 The Properties for Atomic Measurement are as defined in Table 17.

2077 **Table 17 – Properties for Atomic Measurement (in addition to Common Properties defined**  
2078 **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

2079

## 7.9 Query Parameters

### 7.9.1 Introduction

A query string is a fundamental part of the definition of a URI (see 6.2.2). The definition of a query may include Properties and Link Parameters by declaring the Property or Link Parameter (i.e. <Property name, Link Parameter name> = <desired Property value, Link Parameter value>) as one of the segments of the query. Only ASCII strings are permitted in queries, and NULL characters are disallowed in queries. This means that only Property and Link Parameter values with ASCII characters may be matched in a query.

When a query is defined as a selector, a Resource is selected when all the declared Properties or Link Parameters in the query match the corresponding Properties or Link Parameters in the target.

The processing of any query parameter by a Server is as specified in this document or other OCF specifications. For any query parameters that are not explicitly specified, the Server may ignore those query parameters and the request is processed as if the query parameter did not exist in the request.

### 7.9.2 Use of multiple parameters within a query

When a query contains multiple separate query parameters these are delimited by an "&" as described in 6.2.2. Multiple query parameters are only applicable to Collections or Resources with a multi-value "rt".

A Client may select a specific Resource type using separate query parameters, for example "?if=oic.if.b&rt=oic.r.switch.binary". If such queries are supported by the Server this shall be accomplished by matching "all of" the different query parameter types received (i.e. "rt", "if") against the target of the query. In the example, this resolves to a batch response that includes only instances of oic.r.switch.binary. There is no significance applied to the order of the query parameters.

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

A Client may select multiple Resource Types using multiple repeated "rt" parameters in addition to a separate "if" parameter in a single query, for example "?if=oic.if.b&rt=oic.r.switch.binary&rt=oic.r.ramptime". If such queries are supported by the Server, this shall be accomplished by matching "any of" the repeated query parameters and then matching "all of" the different query parameter types. In the example any instances of "oic.r.switch.binary" and/or "oic.r.ramptime" that may exist are selected in a batch response.

NOTE The parameters within a query string are represented within the actual messaging protocol as defined in clause 12.2.2.

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

An "rt" query for a multi-value "rt" Resource with the Default OCF Interface of "oic.if.a", "oic.if.s", "oic.if.r", "oic.if.rw" or "oic.if.baseline" is an extension of a generic "rt" query.

When a Server receives a RETRIEVE request for a multi-value "rt" Resource with an "rt" query, (i.e. GET /ResExample?rt=oic.r.foo), the Server should respond only when the query value is an item of the "rt" Property Value of the target Resource and should send back only the Properties associated with the query value(s). For example, upon receiving GET /ResExample?rt=oic.r.switch.binary targeting a Resource with "rt": ["oic.r.switch.binary", "oic.r.light.brightness"], the Server responds with only the Properties of oic.r.switch.binary.



2126 When a Server receives an UPDATE request for a multi-value "rt" Resource with an "rt" query,  
2127 (e.g. POST /ResExample?rt=oic.r.foo), the Server should only apply the payload received to the  
2128 Properties that are part of the "oic.r.foo" Resource.

## 2129 **7.9.4 OCF Interface specific considerations for queries**

### 2130 **7.9.4.1 OCF Interface selection**

2131 When an OCF Interface is to be selected for a request, it shall be specified as a query parameter  
2132 in the URI of the Resource in the request message. If no query parameter is specified, then the  
2133 Default OCF Interface shall be used. If the selected OCF Interface is not one of the permitted OCF  
2134 Interfaces on the Resource, then selecting that OCF Interface is an error and the Server shall  
2135 respond with an error response code. A Client shall not include more than one OCF Interface in a  
2136 query parameter. If a Server receives a request that has more than one OCF Interface included in  
2137 a query parameter (e.g. "?if=oic.if.ll&if=oic.if.rw") then the Server may either reject the request with  
2138 an appropriate non-success path response, or the Server may attempt to process the request using  
2139 the first "if" received

2140 For example, the baseline OCF Interface may be selected by adding "if=oic.if.baseline" to the list  
2141 of query parameters in the URI of the target Resource. For example: "GET  
2142 /oic/res?if=oic.if.baseline".

### 2143 **7.9.4.2 Batch OCF Interface**

2144 See 7.6.3.4 for details on the batch OCF Interface itself. Query parameters may be used with the  
2145 batch OCF Interface in order to select particular Resources in a Collection for retrieval or update;  
2146 these parameters are used to select items in the Collection by matching Link Parameter Values.

2147 When Link selection query parameters are used with RETRIEVE operations applied using the batch  
2148 OCF Interface, only the Resources in the Collection with matching Link Parameters should be  
2149 returned.

2150 When Link selection query parameters are used with UPDATE operations applied using the batch  
2151 OCF Interface, only the Resources having matching Link Parameters should be updated.

2152 See 7.6.3.4.5 for examples of RETRIEVE and UPDATE operations that use Link selection query  
2153 parameters.

## 2154 **7.10 Error response payload**

### 2155 **7.10.1 Overview**

2156 Clause 7.10 describes a mechanism and payload to signal additional error information that may be  
2157 provided in addition to the response code when an error response is sent. The transport specific  
2158 response for a transport binding (e.g., CoAP) returns a status code that does not always provide  
2159 enough information on what has gone wrong.

### 2160 **7.10.2 Error response payload content**

2161 The error response payload shall be an ASCII string that contains a brief, human-readable  
2162 diagnostic description as a string describing the details of the transport specific error response  
2163 code. Standardized messages for the error response payload are defined in Table 26. Vendors  
2164 may use these standardized messages or define their own messages. The messages contained  
2165 within an error response payload may be included with any transport specific response code.  
2166 English text is the only language supported for the message. If the error response payload is not  
2167 present in the response, a Client deals with the error based on only the transport specific response  
2168 code.

**Table 18 – Standardized error message**

Category	Message
Error due to Client	"Invalid parameter"
	"The mandatory parameter is missing"
	"The parameter is not allowed"
	"The token syntax is invalid"
	"The message id syntax is invalid"
	"Invalid permission"
	"The service key is invalid"
	"The token is not issued"
	"The token user is not issued"
	"Terms of service are not agreed"
	"The API is not permitted"
	"The API call count is exceeded"
	"The country is not supported"
	"The Device is inaccessible"
	"The token is invalid"
	"The count of subscription has exceeded the limit"
	"Invalid resource access"
	"The admin is not registered"
	"The user is not registered"
	"The service is not registered"
	"The event is not subscribed"
	"The Device is not registered"
	"The admin is already registered."
	"Internal Server operation error"
	"Device profile error"
	"The model is not supported"
	"Undefined enumeration"
	"The value is out of range"
	"Feature is not supported in the model"
	"Integration Server error"
	"The product is not supported for interworking with other companies"
	"The Device status is abnormal"
	"The Device is not connected (offline)"
	"The Device control failed"
	"The request is required to retry"
	"Time out occurred"
Error due to Server	"Internal Server operation error"
	"Device profile error"

	"The model is not supported"
	"Undefined enumeration"
	"The value is out of range"
	"Feature is not supported in the model"
	"Integration Server error"
	"The product is not supported for interworking with other companies"
	"The Device status is abnormal"
	"The Device is not connected (offline)"
	"The Device control failed"
	"The request is required to retry"
	"Time out occurred"

2170

2171 **7.10.3 Example of use**

2172 The following example shows an example message exchange for a RETRIEVE operation sent from

2173 a proximal Device to an OCF Cloud, with a target URI of:

2174 "coaps+tcp://exampleCloudEndPoint/deviceId\_001/somehref".

2175 **Client request:**

2176 Target URI: /deviceId\_001/somehref

2177 Operation: RETRIEVE

2178 Host: coaps://exampleCloudEndPoint

2179 Accept: application/vnd.ocf+cbor

2180 **Server response:**

2181 Status code: 4.04 (Not Found)

2182 Response Body: {

2183     "The device is not registered"

2184 }

2185 With the error response payload, the Client can recognize that the Device it tried to discover is not

2186 registered on the OCF Cloud.

## 2187 **8 CRUDN**

### 2188 **8.1 Overview**

2189 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for

2190 manipulating Resources. These operations are performed by a Client on the Resources contained

2191 in a Server. All required Properties shall be present in the payloads for which they are defined for

2192 the operations for which those payloads apply (see clause 7.1 regarding OpenAPI 2.0 definitions

2193 requirement).

2194 On reception of a valid CRUDN operation a Server hosting the Resource that is the target of the

2195 request shall generate a response depending on the OCF Interface included in the request; or

2196 based on the Default OCF Interface for the Resource Type if no OCF Interface is included.

2197 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in

2198 Table 19. A Device shall use CBOR as the default payload (content) encoding scheme for Resource

2199 representations included in CRUDN operations and operation responses; a Device may negotiate

2200 a different payload encoding scheme (e.g, see in 12.2.4 for CoAP messaging). Clauses 8.2 through

8.6 respectively specify the CRUDN operations and use of the parameters. The type definitions for these terms will be mapped in the clause 12 for each protocol.

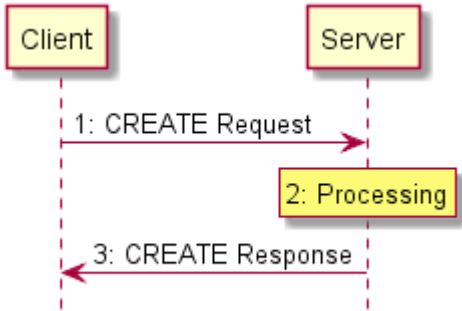
**Table 19 – Parameters of CRUDN messages**

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.

## 8.2 CREATE

### 8.2.1 Overview

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



**Figure 5 – CREATE operation**

### 8.2.2 CREATE request

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

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

### 8.2.3 Processing by the Server

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

### 8.2.4 CREATE response

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

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

## 8.3 RETRIEVE

### 8.3.1 Overview

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

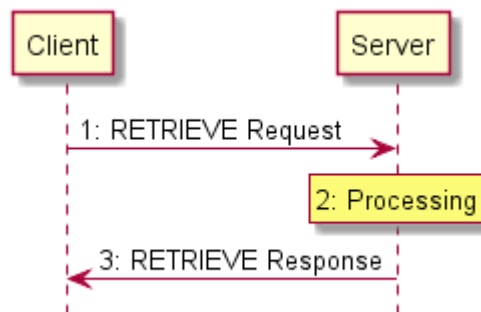


Figure 6 – RETRIEVE operation

### 8.3.2 RETRIEVE request

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

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

### 8.3.3 Processing by the Server

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

#### 8.3.4 RETRIEVE response

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

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

### 8.4 UPDATE

#### 8.4.1 Overview

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

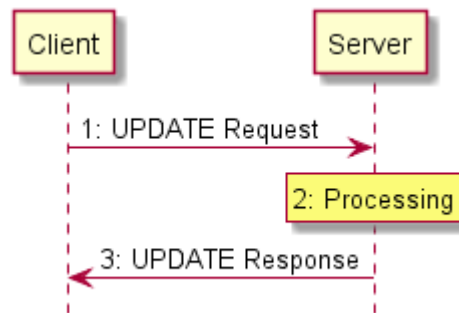


Figure 7 – UPDATE operation

#### 8.4.2 UPDATE request

The UPDATE request message is transmitted by the Client to the Server to request the update of information of a Resource on the Server. The UPDATE request message, as indicated in 8.1, contains all required Properties whether changed or not. The UPDATE request message will carry the following parameters:

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

#### 8.4.3 Processing by the Server

##### 8.4.3.1 Overview

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

2285 An UPDATE request that includes Properties that are read-only shall be rejected by the Server with  
2286 an *rs* indicating a bad request.

2287 An UPDATE request shall be applied only to the Properties in the target Resource visible via the  
2288 applied OCF Interface that support the operation. An UPDATE of non-existent Properties is ignored.

2289 An UPDATE request shall be applied to the Properties in the target Resource even if those Property  
2290 Values are the same as the values currently exposed by the target Resource.

#### 2291 **8.4.3.2 Resource monitoring by the Server**

2292 The Server shall monitor the state the Resource identified in the Observe request from the Client.  
2293 Anytime there is a change in the state of the Observed Resource or an UPDATE operation applied  
2294 to the Resource, the Server sends another RETRIEVE response with the Observe indication. The  
2295 mechanism does not allow the Client to specify any bounds or limits which trigger a notification,  
2296 the decision is left entirely to the Server.

#### 2297 **8.4.3.3 Additional RETRIEVE responses with Observe indication**

2298 The Server shall transmit updated RETRIEVE response messages following Observed changes in  
2299 the state of the Resources requested by the Client. The RETRIEVE response message shall include  
2300 the parameters listed in 11.3.2.4.

#### 2301 **8.4.4 UPDATE response**

2302 The UPDATE response message will include the following parameters:

- 2303 – *fr*: Unique identifier of the Server.
- 2304 – *to*: Unique identifier of the Client.
- 2305 – *ri*: Identifier included in the UPDATE request.
- 2306 – *rs*: The result of the UPDATE request.

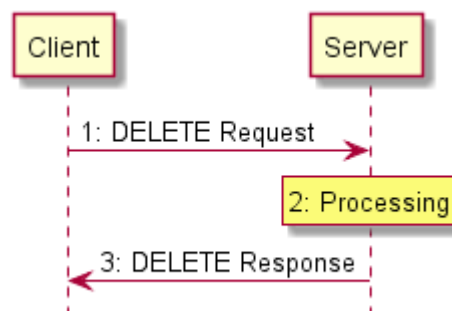
2307 The UPDATE response message may also include the following parameters:

- 2308 – *cn*: The Resource representation following processing of the UPDATE request.

### 2309 **8.5 DELETE**

#### 2310 **8.5.1 Overview**

2311 The DELETE operation is used to request the removal of a Resource. The DELETE operation is  
2312 initiated by the Client and consists of three steps, as depicted in Figure 8.



2313  
2314 **Figure 8 – DELETE operation**

#### 2315 **8.5.2 DELETE request**

2316 DELETE request message is transmitted by the Client to the Server to delete a Resource on the  
2317 Server. The DELETE request message will carry the following parameters:

- 2318 – *fr*: Unique identifier of the Client.
- 2319 – *to*: URI of the target Resource which is the target of deletion.
- 2320 – *ri*: Identifier of the DELETE request.
- 2321 – *op*: DELETE.

### 2322 **8.5.3 Processing by the Server**

2323 Following the receipt of a DELETE request, the Server may validate if the Client has the appropriate  
2324 rights for deleting the identified Resource, and whether the identified Resource exists. If the  
2325 validation is successful, the Server removes the requested Resource and deletes all the associated  
2326 information. The Server caches the value of *ri* parameter in the DELETE request for use in the  
2327 response.

### 2328 **8.5.4 DELETE response**

2329 The Server shall transmit a DELETE response message in response to a DELETE request message  
2330 from a Client. The DELETE response message will include the following parameters:

- 2331 – *fr*: Unique identifier of the Server.
- 2332 – *to*: Unique identifier of the Client.
- 2333 – *ri*: Identifier included in the DELETE request.
- 2334 – *rs*: The result of the DELETE operation.

## 2335 **8.6 NOTIFY**

### 2336 **8.6.1 Overview**

2337 The NOTIFY operation is used to request asynchronous notification of state changes. Complete  
2338 description of the NOTIFY operation is provided in 0. The NOTIFY operation uses the  
2339 NOTIFICATION response message which is defined here.

### 2340 **8.6.2 NOTIFICATION response**

2341 The NOTIFICATION response message is sent by a Server to notify the URLs identified by the  
2342 Client of a state change. The NOTIFICATION response message carries the following parameters:

- 2343 – *fr*: Unique identifier of the Server.
- 2344 – *to*: URI of the Resource target of the NOTIFICATION message.
- 2345 – *ri*: Identifier included in the CREATE request.
- 2346 – *op*: NOTIFY.
- 2347 – *cn*: The updated state of the Resource.

## 2348 **9 Network and connectivity**

### 2349 **9.1 Introduction**

2350 The Internet of Things is comprised of a wide range of applications which sense and actuate the  
2351 physical world with a broad spectrum of device and network capabilities: from battery powered  
2352 nodes transmitting 100 bytes per day and able to last 10 years on a coin cell battery, to mains  
2353 powered nodes able to maintain Megabit video streams. It is estimated that many 10s of billions of  
2354 IoT devices will be deployed over the coming years.

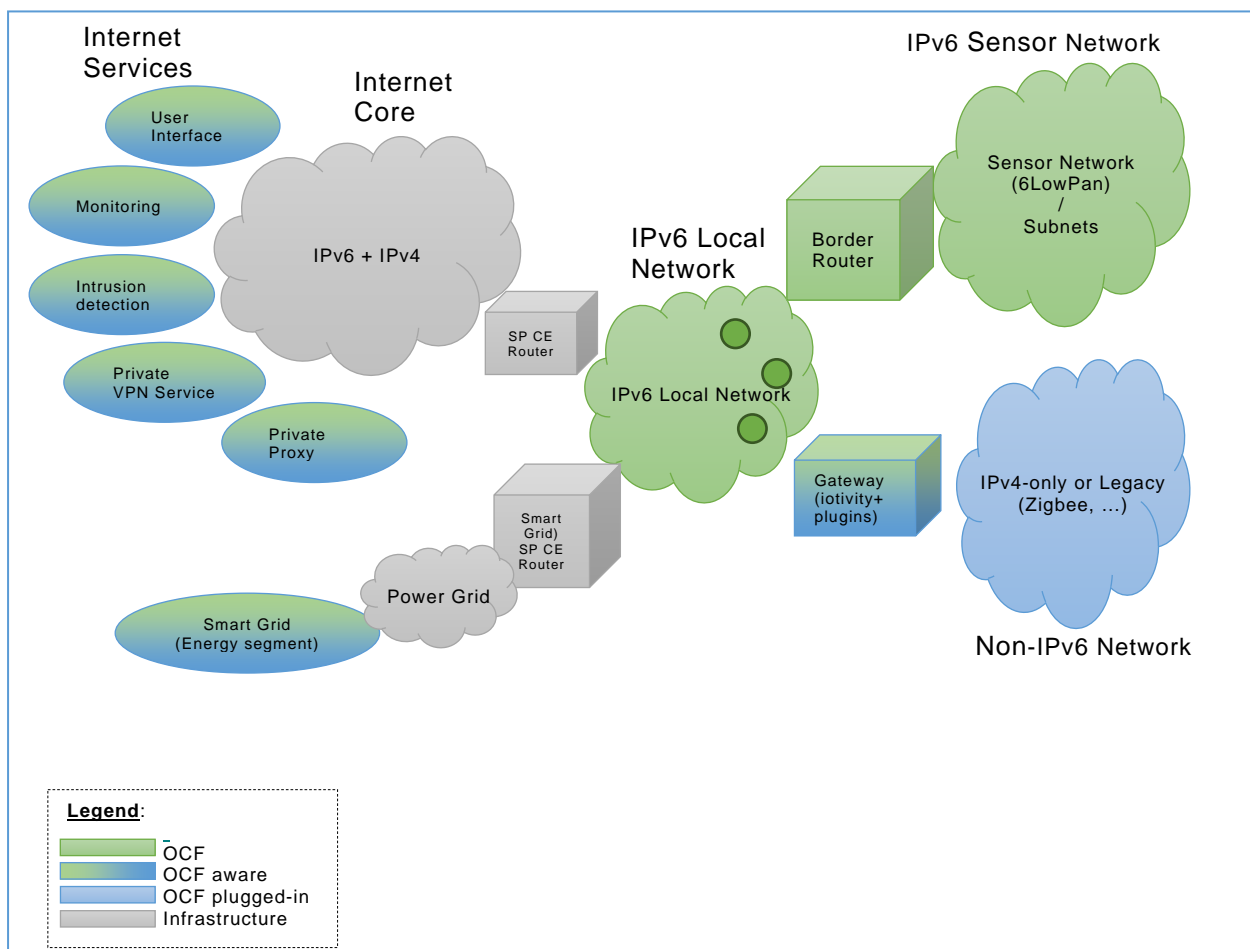
2355 It is desirable that the connectivity options be adapted to the IP layer. To that end, IETF has  
2356 completed considerable work to adapt Bluetooth®, Wi-Fi, 802.15.4, LPWAN, etc. to IPv6. These  
2357 adaptations, plus the larger address space and improved address management capabilities, make  
2358 IPv6 the clear choice for the OCF network layer technology.



## 9.2 Architecture

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

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



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

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

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

2380 – A node may translate and route messaging between IPv6 and non-IPv6 networks.

### 2381 **9.3 IPv6 network layer requirements**

#### 2382 **9.3.1 Introduction**

2383 Projections indicate that many 10s of billions of new IoT endpoints and related services will be  
2384 brought online in the next few years. These endpoint's capabilities will span from battery powered  
2385 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating  
2386 over Ethernet and WiFi links.

2387 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide  
2388 variety of applications such as Web browsing, email, voice, video, and critical system monitoring  
2389 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which  
2390 is that available address space has been consumed.

2391 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF  
2392 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 2393 – Larger address space. Side-effect: greatly reduce the need for NATs.
- 2394 – More flexible addressing architecture. Multiple addresses and types per interface: Link-local,  
2395 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed  
2396 networks, better re-numbering capability, etc.
- 2397 – More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 2398 – Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 2399 – All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
- 2400 – Major Service Providers around the globe are deploying IPv6.

#### 2401 **9.3.2 IPv6 node requirements**

##### 2402 **9.3.2.1 Introduction**

2403 In order to ensure network layer services interoperability from node to node, mandating a common  
2404 network layer across all nodes is vital. The protocol should enable the network to be: secure,  
2405 manageable, and scalable and to include constrained and self-organizing meshed nodes. OCF  
2406 mandates IPv6 as the common network layer protocol to ensure interoperability across all Devices.  
2407 More capable Devices may also include additional protocols creating multiple-stack Devices. The  
2408 remainder of this clause will focus on interoperability requirements for IPv6 hosts, IPv6 constrained  
2409 hosts and IPv6 routers. The various protocol translation permutations included in multi-stack  
2410 gateway devices may be addresses in subsequent addendums of this document.

##### 2411 **9.3.2.2 IP Layer**

2412 An IPv6 node shall support IPv6 and it shall conform to the requirements as specified in  
2413 IETF RFC 6434.

## 2414 **10 OCF Endpoint**

### 2415 **10.1 OCF Endpoint definition**

2416 The specific definition of an OCF Endpoint depends on the Transport Protocol Suite being used.  
2417 For the example of CoAP over UDP over IPv6, the OCF Endpoint is identified by an IPv6 address  
2418 and UDP port number.

2419 Each Device shall associate with at least one OCF Endpoint with which it can exchange request  
2420 and response messages. When a message is sent to an OCF Endpoint, it shall be delivered to the  
2421 Device which is associated with the OCF Endpoint. When a request message is delivered to an  
2422 OCF Endpoint, path component is enough to locate the target Resource.

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

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

## 10.2 OCF Endpoint information

### 10.2.1 Introduction

An OCF Endpoint is represented by OCF Endpoint information, which consists of the following key-value pairs, "ep", "pri", and "lat".

### 10.2.2 "ep"

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

- *Transport Protocol Suite* - a combination of protocols (e.g. CoAP + UDP + IPv6) with which request and response messages can be exchanged for RESTful transaction (i.e. CRUDN). A Transport Protocol Suite shall be indicated by a URI scheme name. All scheme names supported by this document are IANA registered, these are listed in Table 20. A vendor may also make use of a non-IANA registered scheme name for their own use (e.g. "com.example.foo"), this shall follow the syntax for such scheme names defined by IETF RFC 7595. The behaviour of a vendor-defined scheme name is undefined by this document. All OCF defined Resource Types when exposing OCF Endpoint Information in an "ep" (see 10.2.4) shall include at least one "ep" with a Transport Protocol Suite as defined in Table 20.
- *OCF Endpoint Locator* – an address (e.g. IPv6 address + Port number) or an indirect identifier (e.g., DNS name) resolvable to an IP address, through which a message can be sent to the OCF Endpoint and in turn associated Device. The OCF Endpoint Locator for "coap" and "coaps" shall be specified as "IP address: port number". The OCF Endpoint Locator for "coap+tcp" or "coaps+tcp" shall be specified as "IP address: port number" or "DNS name: port number" or "DNS name" such that the DNS name shall be resolved to a valid IP address for the target Resource with a name resolution service (i.e., DNS). For the 3rd case, when the port number is omitted, the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and for "coaps+tcp") scheme respectively as defined in IETF RFC 8323. Temporary addresses should not be used because OCF Endpoint Locators are for the purpose of accepting incoming sessions, whereas temporary addresses are for initiating outgoing sessions (IETF RFC 4941). Moreover, its inclusion in "/oic/res" can cause a privacy concern (IETF RFC 7721).
- *OCF Latency* – the maximum latency in seconds [sec] that the Server may take to respond to a request.

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

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

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

2468 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"
```

2469 The current list of "ep" with corresponding Transport Protocol Suite is shown in Table 20:

2470 **Table 20 – "ep" value for Transport Protocol Suite**

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

2471

### 2472 10.2.3 "pri"

2473 When there are multiple OCF Endpoints, "pri" indicates the priority among them.

2474 "pri" shall be represented as a positive integer (e.g. "pri": 1) and the lower the value, the higher the  
2475 priority.

2476 The default "pri" value is 1, i.e. when "pri" is not present, it shall be equivalent to "pri": 1.

### 2477 10.2.4 "lat"

2478 "lat" indicates the expected delay of the response. For example, when a Server implements a mode  
2479 to improve battery performance; the Server can expose this value, thereby providing a Client with  
2480 the ability to use this for the timeout on the connection. For example, the Thread "rx-off-when-idle"  
2481 link mode is an implementation of a battery performance improvement mechanism.

2482 "lat" shall be represented as a positive integer (e.g. "lat": 240), and the value is specified in seconds.

### 2483 10.2.5 OCF Endpoint information in "eps" Parameter

2484 To carry OCF Endpoint information, a new Link Parameter "eps" is defined in 7.8.2.5.6. "eps" has  
2485 an array of items as its value and each item represents OCF Endpoint information with key-value  
2486 pairs, "ep", "pri", and "lat", of which "ep" is mandatory and "pri" and "lat" are optional.

2487 OCF Endpoint Information in an "eps" Parameter is valid for the target Resource of the Link, i.e.,  
2488 the Resource referred by "href" Parameter. OCF Endpoint information in an "eps" Parameter may  
2489 be used to access other Resources on the Device, but such access is not guaranteed.

2490 A Client may resolve the "ep" value to an IP address for the target Resource, i.e., the address to  
2491 access the Device which hosts the target Resource. A valid (transfer protocol) URI for the target  
2492 Resource can be constructed with the scheme, host and port components from the "ep" value and  
2493 the "path" component from the "href" value.

2494 Links with an "eps":

```
2495 {  
2496   "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9 ",  
2497   "href": "/myLightSwitch",  
2498   "rt": ["oic.r.switch.binary"],  
2499   "if": ["oic.if.a", "oic.if.baseline"],  
2500   "p": {"bm": 3},  
2501   "eps": [  
2502     {"ep": "coap://[fe80::b1d6]:1111", "pri": 2, "lat": 240},  
2503     {"ep": "coaps://[fe80::b1d6]:1122"}  
2504   ]  
2505 }  
2506  
2507 {  
2508   "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",  
2509   "href": "/myTemperature",  
2510   "rt": ["oic.r.temperature"],  
2511   "if": ["oic.if.a", "oic.if.baseline"],  
2512   "p": {"bm": 3},  
2513   "eps": [  
2514     {"ep": "coap+tcp://foo.bar.com", "pri": 2, "lat": 240},  
2515     {"ep": "coaps+tcp://foo.bar.com:1122"}  
2516   ]  
2517 }
```

2518 In the previous example, "anchor" represents the hosting Device, "href", target Resource and "eps"  
2519 the two OCF Endpoints for the target Resource. The (fully-qualified) URIs for the target Resource  
2520 are as illustrated:

```
2521 coap://[fe80::b1d6]:1111/myLightSwitch  
2522 coaps://[fe80::b1d6]:1122/myLightSwitch  
2523 coap+tcp://foo.bar.com:5683/myTemperature
```

2524 coaps+tcp://foo.bar.com:1122/myTemperature If the target Resource of a Link requires a secure  
2525 connection (e.g. CoAPS), "eps" Parameter shall be used to indicate the necessary information (e.g.  
2526 port number) in OCF 1.0 payload. For optional backward compatibility with OIC 1.1, the "sec" and  
2527 "port" shall only be used in OIC 1.1 payload.

## 2528 10.3 OCF Endpoint discovery

### 2529 10.3.1 Introduction

2530 OCF Endpoint discovery is defined as the process for a Client to acquire the OCF Endpoint  
2531 information for Device or Resource.

### 2532 10.3.2 Implicit discovery

2533 If a Device is the source of a CoAP message (e.g. "/oic/res" response), the source IP address and  
2534 port number may be combined to form the OCF Endpoint Locator for the Device. Along with a  
2535 "coap" scheme and default "pri" value, OCF Endpoint information for the Device may be constructed.

2536 In other words, a "/oic/res" response message with CoAP may implicitly carry the OCF Endpoint  
2537 information of the responding Device and in turn all the hosted Resources, which may be accessed  
2538 with the same transfer protocol of CoAP. In the absence of an "eps" Parameter, a Client shall be  
2539 able to utilize implicit discovery to access the target Resource.

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

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

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

When the OCF Endpoint for a target Resource of a Link cannot be implicitly inferred, the "eps" Parameter shall be included to provide explicit OCF Endpoint information with which a Client can access the target Resource. In the presence of the "eps" Parameter, a Client shall be able to utilize it to access the target Resource. For "coap" and "coaps", a Client may use the IP address in the "ep" value in the "eps" Parameter to access the target Resource. For "coap+tcp" and "coaps+tcp", a Client may use the IP address in the "eps" Parameter or resolve the DNS name in the "eps" Parameter to acquire a valid IP address for the target Resource. If "eps" Parameter omits the port number, then the default port "5683" (and "5684") shall be assumed for "coap+tcp" (and "coaps+tcp") scheme as defined in IETF RFC 8323. To access the target Resource of a Link, a Client may use the "eps" Parameter in the Link, if it is present and fall back on implicit discovery if not.

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

```
[
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/res",
    "rel": "self",
    "rt": ["oic.wk.res"],
    "if": ["oic.if.ll", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/d",
    "rt": ["oic.wk.d"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  },
  {
    "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
    "href": "/oic/p",
    "rt": ["oic.wk.p"],
    "if": ["oic.if.r", "oic.if.baseline"],
    "p": {"bm": 3},
    "eps": [
      {"ep": "coap://[2001:db8:a::b1d4]:55555"},
      {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
    ]
  }
]
```

```

2596     },
2597     {
2598         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2599         "href": "/oic/sec/doxm",
2600         "rt": ["oic.r.doxm"],
2601         "if": ["oic.if.baseline"],
2602         "p": {"bm": 1},
2603         "eps": [
2604             {"ep": "coap://[2001:db8:a::b1d4]:55555"},
2605             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2606         ]
2607     },
2608     {
2609         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2610         "href": "/oic/sec/pstat",
2611         "rt": ["oic.r.pstat"],
2612         "if": ["oic.if.baseline"],
2613         "p": {"bm": 1},
2614         "eps": [
2615             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2616         ]
2617     },
2618     {
2619         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2620         "href": "/oic/sec/cred",
2621         "rt": ["oic.r.cred"],
2622         "if": ["oic.if.baseline"],
2623         "p": {"bm": 1},
2624         "eps": [
2625             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2626         ]
2627     },
2628     {
2629         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2630         "href": "/oic/sec/acl2",
2631         "rt": ["oic.r.acl2"],
2632         "if": ["oic.if.baseline"],
2633         "p": {"bm": 1},
2634         "eps": [
2635             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2636         ]
2637     },
2638     {
2639         "anchor": "ocf://e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
2640         "href": "/myIntrospection",
2641         "rt": ["oic.wk.introspection"],
2642         "if": ["oic.if.r", "oic.if.baseline"],
2643         "p": {"bm": 3},
2644         "eps": [
2645             {"ep": "coaps://[2001:db8:a::b1d4]:11111"}
2646         ]
2647     },
2648     {
2649         "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2650         "href": "/myLight",
2651         "rt": ["oic.r.switch.binary"],
2652         "if": ["oic.if.a", "oic.if.baseline"],
2653         "p": {"bm": 3},
2654         "eps": [
2655             {"ep": "coaps://[2001:db8:a::b1d4]:22222"}
2656         ]
2657     }
2658 ]

```

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

## 11 Functional interactions

### 11.1 Introduction

The functional interactions between a Client and a Server are described in 11.1 through 11.4 respectively. The functional interactions use CRUDN messages (clause 8) and include Discovery, Notification, and Device management. These functions require support of core defined Resources as defined in Table 21.

**Table 21 – List of Core Resources**

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

### 11.2 Resource discovery

#### 11.2.1 Introduction

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

#### 11.2.2 Resource based discovery: mechanisms

##### 11.2.2.1 Overview

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

At the minimum, Resource based discovery uses the following:

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



2692 Depending on the choice of the base aspects, the Framework defines three Resource based  
2693 discovery mechanisms:

- 2694 – Direct discovery, where the Resources are published locally at the Device hosting the  
2695 Resources and are discovered through peer inquiry.
- 2696 – Indirect discovery, where Resources are published at a third party assisting with the discovery  
2697 and peers publish and perform discovery against the Resource to enable discovery on the  
2698 assisting 3<sup>rd</sup> party.
- 2699 – Advertisement discovery, where the Resource to enable discovery is hosted local to the initiator  
2700 of the discovery inquiry but remote to the Devices that are publishing discovery information.

2701 A Device shall support direct discovery.

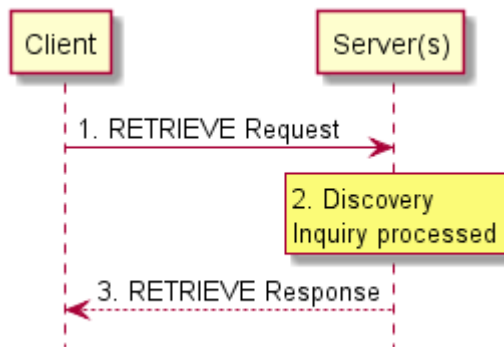
#### 2702 **11.2.2.2 Direct discovery**

2703 In direct discovery,

- 2704 – The Device that is providing the information shall host the Resource to enable discovery.
- 2705 – The Device publishes the information available for discovery with the local Resource to enable  
2706 discovery (i.e. local scope).
- 2707 – Clients interested in discovering information about this Device shall issue RETRIEVE requests  
2708 directly to the Resource. The request may be made as a unicast or multicast. The request may  
2709 be generic or may be qualified or limited by using appropriate queries in the request.
- 2710 – The Server Device that receives the request shall send a response with the discovered  
2711 information directly back to the requesting Client Device.
- 2712 – The information that is included in the request is determined by the policies set for the Resource  
2713 to be discovered locally on the responding Device.

#### 2714 **11.2.3 Resource based discovery: Finding information**

2715 The discovery process (Figure 10) is initiated as a RETRIEVE request to the Resource to enable  
2716 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as  
2717 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the  
2718 support in the data connectivity layer. The response to the request has the information to be  
2719 discovered based on the policies for that information. The policies can determine which information  
2720 is shared, when and to which requesting agent. The information that can be discovered can be  
2721 Resources, types, configuration and many other standards or custom aspects depending on the  
2722 request to appropriate Resource and the form of request. Optionally the requester may narrow the  
2723 information to be returned in the request using query parameters in the URI query.



2724

2725 **Figure 10 – Resource based discovery: Finding information**

2726

2727 *Discovery Resources*

2728 The following Core Resources shall be implemented on all Devices to support discovery:

2729 – "/oic/res" for discovery of Resources.

2730 – "/oic/p" for discovery of Platform.

2731 – "/oic/d" for discovery of Device information.

2732 Devices shall expose each of "/oic/res", "/oic/d", and "/oic/p" via an unsecured OCF Endpoint.  
2733 Further details for these mandatory Core Resources are described in Table 22.

2734 *Platform Resource*

2735 The OCF recognizes that more than one instance of Device may be hosted on a single Platform.  
2736 Clients need a way to discover and access the information on the Platform. The Core Resource,  
2737 "/oic/p" exposes Platform specific Properties. All instances of Device on the same Platform shall  
2738 have the same values of any Properties exposed (i.e. a Device may choose to expose optional  
2739 Properties within "/oic/p" but when exposed the value of that Property should be the same as the  
2740 value of that Property on all other Devices on that Platform).

2741 *Device Resource*

2742 The Device Resource shall have the pre-defined URI "/oic/d", the Device Resource shall expose  
2743 the Properties pertaining to a Device as defined in Table 25. The Device Resource shall have a  
2744 default Resource Type that helps in bootstrapping the interactions with the Device (the default type  
2745 is described in Table 22). The Device Resource may have one or more Resource Type(s) that are  
2746 specific to the Device in addition to the default Resource Type or if present overriding the default  
2747 Resource Type. The base Resource Type "oic.wk.d" defines the Properties that shall be exposed  
2748 by all Devices. The Device specific Resource Type(s) exposed are dependent on the class of  
2749 Device (e.g. air conditioner, smoke alarm, etc. Since all the Resource Types of "/oic/d" are not  
2750 known a priori, the Resource Type(s) of "/oic/d" are determined by discovery through the Core  
2751 Resource "/oic/res".

2752 **Table 22 – Mandatory discovery Core Resources**

Pre-defined URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
<b>"/oic/res"</b>	Default	"oic.wk.res"	"oic.if.ll", "oic.if.b", "oic.if.baseline"	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 23.	Discovery
<b>"/oic/p"</b>	Platform	"oic.wk.p"	"oic.if.r"	The Discoverable Resource through which Platform specific information is discovered.	Discovery

				The Properties exposed by "/oic/p" are listed in Table 26	
<b>"/oic/d"</b>	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 25.	Discovery

2753 Table 23 defines "oic.wk.res" Resource Type.

2754 **Table 23 – "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.
Security Domain UUID	"sduuid"	string	uuid	N/A	R	No	Unique identifier for the Security Domain. This value shall be the same value (i.e. mirror) as the "sdi.uuid" Property as defined in ISO/IEC 30118-2. It shall be exposed if the "sdi.priv" Property is set to "false", and shall not be exposed if the "sdi.priv" Property is set to "true".
Security Domain Name	"sdname"	string	N/A	N/A	R	No	Human-friendly name for the Security Domain. This value shall be the same value (i.e. mirror) as the "sdi.name" Property as defined in ISO/IEC 30118-2. It shall be exposed if the "sdi.priv" Property is set to "false", and shall not be exposed if the "sdi.priv" Property is set to "true".

2755 Note: The "n", "sduuid", and "sdname" Property values for the "oic.wk.res" Resource Type are only in the response  
2756 payload when used with the "oic.if.baseline" OCF Interface (i.e., RETRIEVE /oic/res?if="oic.if.baseline").

2757 A Device shall support CoAP based discovery as the baseline discovery mechanism (see 11.2.5).

2758 The "/oic/res" shall list all Resources that are indicated as discoverable (see 11.2). Also the  
2759 following architecture Resource Types shall be listed:

- 2760 – Introspection Resource indicated with an "rt" value of "oic.wk.introspection".
- 2761 – "/oic/p" indicated with an "rt" value of "oic.wk.p".
- 2762 – "/oic/d" indicated with an "rt" value of "oic.wk.d"

- 2763 – "/oic/sec/doxm" indicated with an "rt" value of "oic.r.doxm" as defined in ISO/IEC 30118-2.
- 2764 – "/oic/sec/pstat" indicated with an "rt" value of "oic.r.pstat" as defined in ISO/IEC 30118-2.
- 2765 – "/oic/sec/acl2" indicated with an "rt" value of "oic.r.acl2" as defined in ISO/IEC 30118-2.
- 2766 – "/oic/sec/cred" indicated with an "rt" value of "oic.r.cred" as defined in ISO/IEC 30118-2.

2767 Conditionally required:

- 2768 – "/oic/res" with an "rt" value of "oic.wk.res" as self-reference, on the condition that "oic/res" has  
2769 to signal that it is Observable by a Client.
- 2770 – if the Device supports batch retrieval of "/oic/res" then "oic.if.b" shall be included in the "if"  
2771 Property of "/oic/res".
- 2772 – if the Device supports batch retrieval there shall be a self-reference that includes an "if" Link  
2773 Parameter containing "oic.if.b"; the self-reference shall expose a secure OCF Endpoint.

2774 The Introspection Resource is only applicable for Devices that host Vertical Resource Types (e.g.  
2775 "oic.r.switch.binary") or vendor-defined Resource Types. Devices that only host Resources  
2776 required to onboard the Device as a Client do not have to implement the Introspection Resource.

2777 Table 24 provides an OCF registry for protocol schemes.

2778 **Table 24 – Protocol scheme registry**

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

2779

2780 NOTE The discovery of an OCF Endpoint used by a specific protocol is out of scope. The mechanism used by a Client  
2781 to form requests in a different messaging protocol other than discovery is out of scope.

2782 The following applies to the use of "/oic/d":

- 2783 – A vertical may choose to extend the list of Properties defined by the Resource Type "oic.wk.d".  
2784 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The  
2785 mandatory Properties defined in Table 25 shall always be present.
- 2786 – A Device may choose to expose a separate, Discoverable Resource with its Resource Type ID  
2787 set to a Device Type. In this case the Resource is equivalent to an instance of "oic.wk.d" and  
2788 adheres to the definition thereof. As such the Resource shall at a minimum expose the  
2789 mandatory Properties of "oic.wk.d". In the case where the Resource tagged in this manner is  
2790 defined to be an instance of a Collection in accordance with 7.8.3 then the Resources that are  
2791 part of that Collection shall at a minimum include the Resource Types mandated for the Device  
2792 Type.

2793 Table 25 "oic.wk.d" Resource Type definition defines the base Resource Type for the "/oic/d"  
2794 Resource.

**Table 25 – "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 UUID	"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. Handling privacy-sensitivity for the "di" Property, refer to clause 13.16 in ISO/IEC 30118-2.
Data Model Version	"dmv"	"csv"	N/A	N/A	R	Yes	Spec version of the Resource specification to which this Device data model is implemented; if implemented against a Vertical specific Device specification(s), then the Spec version of the vertical specification this Device model is implemented to. The syntax is a comma separated list of <res>.<major>.<minor>.<sub-version> or <vertical>.<major>.<minor>.<sub-version>. <res> is the string "ocf.res" and <vertical> is the name of the vertical defined in the Vertical specific Resource specification. The <major>, <minor>, and <sub-version> are the major, minor and sub-version numbers of the specification respectively. One entry in the csv string shall be the applicable version of the Resource Type Specification for the Device (e.g. "ocf.res.1.0.0"). If applicable, additional entry(-ies) in the csv shall be the vertical(s) being realized (e.g. "ocf.sh.1.0.0"). This value may be extended by the vendor. The syntax for extending this value, as a comma separated

							entry, by the vendor shall be by adding x.<Domain_Name>.<vendor_string> . For example, "ocf.res.1.0.0, ocf.sh.1.0.0, x.com.example.string", The order of the values in the comma separated string can be in any order (i.e. no prescribed order). This Property shall not exceed 256 octets.
Permanent Immutable ID	"piid"	"uuid"	N/A	N/A	R	Yes	A unique and immutable Device identifier. A Client can detect that a single Device supports multiple communication protocols if it discovers that the Device uses a single Permanent Immutable ID value for all the protocols it supports. Handling privacy-sensitivity for the "piid" Property, refer to clause 13.16 in ISO/IEC 30118-2.
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.

2796 Table 26 defines "oic.wk.p" Resource Type.

**Table 26 – "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.
<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.
<b>Network Connectivity Type</b>	"mnnci"	"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.
--	--	--	--	--	--	--	---

## 11.2.4 Resource discovery using "/oic/res"

### 11.2.4.1 General Requirements

Discovery using "/oic/res" is the default discovery mechanism that shall be supported by all Devices. General requirements for use of this mechanism are as follows:

- Every Device updates its local "/oic/res" with the Resources that are discoverable (see 7.3.2.2). Every time a new Resource is instantiated on the Device and if that Resource is discoverable by a remote Device then that Resource is published with the "/oic/res" Resource that is local to the Device (as the instantiated Resource).

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

### 11.2.4.2 Discovery using "oic.if.ll" (Default OCF Interface for "/oic/res")

If a Client does not explicitly include an OCF Interface as a query parameter in the request to "/oic/res" then the OCF Interface is taken to be "oic.if.ll" as that is the Default OCF Interface for "/oic/res". The requirements in this clause are thus applied. The requirements in this clause also apply if an OCF Interface of "oic.if.ll" is explicitly requested by inclusion as a query parameter in the RETRIEVE operation.

- A Device wanting to discover Resources or Resource Types on one or more remote Devices makes a RETRIEVE request to the "/oic/res" on the remote Devices. This request may be sent multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request may optionally be restricted using appropriate clauses in the query portion of the request. Queries may select based on Resource Types, OCF Interfaces, or Properties.
- The query applies to the representation of the Resources. "/oic/res" is the only Resource whose representation has "rt". So "/oic/res" is the only Resource that can be used for Multicast discovery at the transport protocol layer.
- The Device receiving the RETRIEVE request responds with a list of Resources, the Resource Type of each of the Resources and the OCF Interfaces that each Resource supports. Additionally, information on the policies active on the Resource can also be sent. The policy supported includes Observability and discoverability.
- The receiving Device may do a deeper discovery based on the Resources returned in the request to "/oic/res".

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

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

For Clients that do include the OCF-Accept-Content-Format-Version option, an "/oic/res" response includes an array of Links to conform to IETF RFC 6690. Each Link shall use an "eps" Parameter



2839 to provide the information for an encrypted connection and carry "anchor" containing an OCF URI  
2840 where the authority component of is the Device UUID of the Device hosting the target Resource.

2841 The OpenAPI 2.0 file for discovery using "/oic/res" is described in Annex A. Also refer to clause 10  
2842 (OCF Endpoint discovery) for details of Multicast discovery using "/oic/res" on a CoAP transport.

2843 An example Device might return the following to Clients that request with the Content Format of  
2844 "application/vnd.ocf+cbor" in Accept Option:

```
2845 [
2846   {
2847     "href": "/oic/res",
2848     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2849     "rel": "self",
2850     "rt": ["oic.wk.res"],
2851     "if": ["oic.if.ll", "oic.if.baseline"],
2852     "p": {"bm": 3},
2853     "eps": [{"ep": "coap://[fe80::b1d6]:44444"}]
2854   },
2855   {
2856     "href": "/oic/p",
2857     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2858     "rt": ["oic.wk.p"],
2859     "if": ["oic.if.r", "oic.if.baseline"],
2860     "p": {"bm": 3},
2861     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2862             {"ep": "coaps://[fe80::b1d6]:11111"}]
2863   },
2864   {
2865     "href": "/oic/d",
2866     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2867     "rt": ["oic.wk.d"],
2868     "if": ["oic.if.r", "oic.if.baseline"],
2869     "p": {"bm": 3},
2870     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2871             {"ep": "coaps://[fe80::b1d6]:11111"}]
2872   },
2873   {
2874     "href": "/myLightSwitch",
2875     "anchor": "ocf://dc70373c-1e8d-4fb3-962e-017eaa863989",
2876     "rt": ["oic.r.switch.binary"],
2877     "if": ["oic.if.a", "oic.if.baseline"],
2878     "p": {"bm": 3},
2879     "eps": [{"ep": "coap://[fe80::b1d6]:44444"},
2880             {"ep": "coaps://[fe80::b1d6]:11111"}]
2881   }
2882 ]
2883
2884
2885
```

### 2886 11.2.5 Multicast discovery using "/oic/res"

2887 Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices shall  
2888 support use of CoAP multicast to allow retrieving the "/oic/res" Resource from an unsecured OCF  
2889 Endpoint on the Device. Clients may support use of CoAP multicast to retrieve the "/oic/res"  
2890 Resource from other Devices. The CoAP multicast retrieval of "/oic/res" supports filtering Links  
2891 based on the "rt" Property in the Links:

- 2892 – If the discovery request is intended for a specific Resource Type including as part of a multi-  
2893 value Resource Type, the query parameter "rt" shall be included in the request (see 6.2.2) with

2894 its value set to the desired Resource Type. Only Devices hosting the Resource Type shall  
2895 respond to the discovery request.

2896 – When the "rt" query parameter is omitted, all Devices shall respond to the discovery request.

## 2897 **11.2.6 Multicast discovery using "/.well-known/core"**

2898 Generic requirements for use of CoAP multicast are provided in clause 12.2.9. Devices that join  
2899 the All CoAP Nodes multicast group as optionally defined in clause 12.2.9 may also support  
2900 multicast retrieval from "/.well-known/core" (see IETF RFC 7252). A Server node shall join at  
2901 least both the link-local scoped address FF02::FD and the site-local scoped address  
2902 FF05::FD. IPv6 addresses of other scopes may also be enabled. A Device responding to a  
2903 request received on "/.well-known/core" shall encode the payload using the Core link format, which  
2904 is a Content-Format of "40" (application/link-format) as defined in IETF RFC 6690. Core links in  
2905 the response payload shall have a Content-Format code ("ct" attribute) of "10000"  
2906 ("application/vnd.ocf+cbor"). This Content-Format code shall be used in subsequent requests and  
2907 responses to obtain further Device Resource information.

2908 A Client may send a multicast request to "/.well-known/core" to discover Devices that have joined  
2909 the All CoAP Nodes multicast group. However, non-OCF Devices may also respond to this request.  
2910 In order to filter out these non-OCF Devices, a Client may use "rt" query parameters so that only  
2911 OCF Devices respond. A Server shall support querying for the "oic.wk.res" Resource Type as an  
2912 "rt" query parameter value. A Client issuing such a request is equivalent to searching for all  
2913 Devices. The Server shall also support querying for a Device Type as an "rt" query parameter value  
2914 and respond when the Device Type matches the "rt" query parameter value.

2915 Devices that support this optional discovery mechanism shall return as a minimum the Core link to  
2916 the "/oic/res" Resource so that discovery of further Resources may be performed with a RETRIEVE  
2917 operation to the URL of the discovered "/oic/res" Resource. The returned URL shall be fully  
2918 qualified.

2919 The "rt" and "if" attribute shall also be included in the response. The "rt" attribute shall include  
2920 "oic.wk.res" and the "rt" value of the Device Type. The "if" attribute shall include the OCF Interfaces  
2921 exposed by "/oic/res".

2922 Example of a query for all Devices:

```
2923 Req: GET coap://[FF02::FD]:5683/.well-known/core?rt=oic.wk.res
2924 Res: 2.05 Content, Content-Format: 40
2925 <coap://[fe80::b1d6]:1111/oic/res>;ct=10000;rt="oic.wk.res oic.d.sensor";if="oic.if.11
2926 oic.if.baseline";
```

2927 Example of a query for a specific Device Type:

```
2928 Req: GET coap://[FF02::FD]:5683/.well-known/core?rt=oic.d.sensor
2929 Res: 2.05 Content, Content-Format: 40
2930 <coap://[fe80::b1d6]:1111/oic/res>;ct=10000;rt="oic.wk.res oic.d.sensor"; if="oic.if.11
2931 oic.if.baseline"
```

## 2932 **11.3 Notification**

### 2933 **11.3.1 Overview**

2934 A Server shall support NOTIFY operation to enable a Client to request and be notified of desired  
2935 states of one or more Resources in an asynchronous manner. 11.3.2 specifies the Observe  
2936 mechanism in which updates are delivered to the requester.

## 11.3.2 Observe

### 11.3.2.1 Overview

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

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

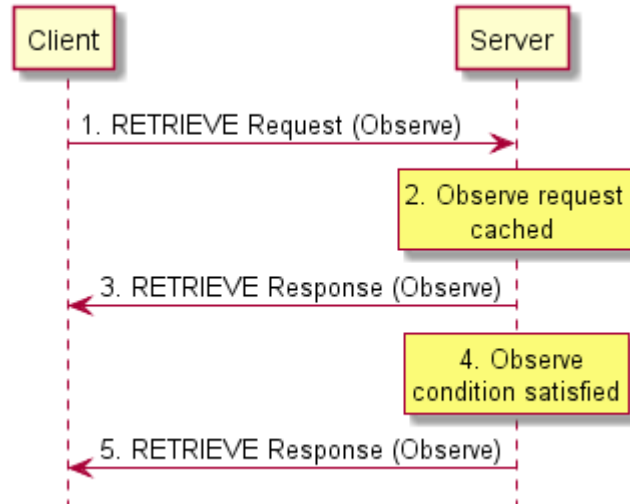


Figure 11 – Observe Mechanism

### 11.3.2.2 RETRIEVE request with Observe indication

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

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

### 11.3.2.3 Processing by the Server

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

### 11.3.2.4 RETRIEVE response with Observe indication

The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request message from a Client. If validation succeeded, the response includes an Observe indication. If

2965 not, the Observe indication is omitted from the response which signals to the requesting Client that  
2966 registration for notification was not allowed.

2967 The RETRIEVE response message shall include the following parameters:

- 2968 – *fr*: Unique identifier of the Server.
- 2969 – *to*: Unique identifier of the Client.
- 2970 – *ri*: Identifier included in the RETRIEVE operation.
- 2971 – *cn*: Information Resource representation as requested by the Client.
- 2972 – *rs*: The result of the RETRIEVE operation.
- 2973 – *obs*: Indication that the response is made to an Observe operation.

#### 2974 **11.3.2.5 Resource monitoring by the Server**

2975 The Server shall monitor the state the Resource identified in the Observe request from the Client.  
2976 Anytime there is a change in the state of the Observed Resource, the Server sends another  
2977 RETRIEVE response with the Observe indication. The mechanism does not allow the client to  
2978 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

#### 2979 **11.3.2.6 Additional RETRIEVE responses with Observe indication**

2980 The Server shall transmit updated RETRIEVE response messages following Observed changes in  
2981 the state of the Resources indicated by the Client. The RETRIEVE response message shall include  
2982 the parameters listed in 11.3.2.4.

#### 2983 **11.3.2.7 Cancelling Observe**

2984 The Client can explicitly cancel Observe by sending a RETRIEVE request without the Observe  
2985 indication field to the same Resource on the Server which it was Observing. For certain protocol  
2986 mappings, the Client may also be able to cancel an Observe by ceasing to respond to the  
2987 RETRIEVE responses.

### 2988 **11.4 Introspection**

#### 2989 **11.4.1 Overview**

2990 Introspection is a mechanism to announce the capabilities of Resources hosted on the Device.

2991 The intended usage of the Introspection Device Data (IDD) is to enable dynamic Clients e.g. Clients  
2992 that can use the IDD) to generate dynamically a UI or dynamically create translations of the hosted  
2993 Resources to another eco-system. Other usages of Introspection is that the information can be  
2994 used to generate Client code. The IDD is designed to augment the existing data already on the  
2995 wire. This means that existing mechanisms need to be used to get a full overview of what is  
2996 implemented in the Device. For example, the IDD does not convey information about Observability,  
2997 since that is already conveyed with the "p" Property on the Links in "/oic/res" (see 7.8.2.5.3).

2998 The IDD is recommended to be conveyed as static data. Meaning that the data does not change  
2999 during the uptime of a Device. However, when the IDD is not static, the Introspection Resource  
3000 shall be Observable and the url Property Value of "oic.wk.introspection" Resource shall change to  
3001 indicate that the IDD is changed.

3002 The IDD describes the Resources that make up the Device. For the complete list of included  
3003 Resources see Table 21. The IDD is described as a OpenAPI 2.0 in JSON format file. The text in  
3004 the following bulleted list contains OpenAPI 2.0 terms, such as paths, methods etc. The OpenAPI  
3005 2.0 file shall contain the description of the Resources:

- 3006 – The IDD will use the HTTP syntax, e.g., define the CRUDN operation as HTTP methods and  
3007 use the HTTP status codes.

- 3008 – The IDD does not have to define all the status codes that indicate an error situation.
- 3009 – The IDD does not have to define a schema when the status code indicates that there is no  
3010 payload (see HTTP status code 204 as an example).
- 3011 – The paths (URLs) of the Resources in the IDD shall be without the OCF Endpoint description,  
3012 e.g. it shall not be a fully-qualified URL but only the relative path from the OCF Endpoint, aka  
3013 the "href". The relative path may include a query parameter (e.g. "?if=oic.if.ll"), in such cases  
3014 the text following (and including) the "?" delimiter shall be removed before equating to the "href"  
3015 that is conveyed by "/oic/res".
- 3016 – The following Resources shall be excluded in the IDD:
  - 3017 – Resource with Resource Type: "oic.wk.res" unless 3rd party defined or optional Properties  
3018 are implemented.
  - 3019 – Resource with Resource Type: "oic.wk.introspection".
  - 3020 – Resources explicitly identified within other specifications working in conjunction with this  
3021 document (e.g. Resources that handle Wi-Fi Easy Setup, see [2]).
- 3022 – The following Resources shall be included in the IDD when optional or 3<sup>rd</sup> party defined  
3023 Properties are implemented:
  - 3024 – Resources with type: "oic.wk.p" and "oic.wk.d" (e.g. discovery related Resources).
  - 3025 – Security Virtual Resources from ISO/IEC 30118-2.
- 3026 – When the Device does not expose instances of Vertical Resource Types, and does not have  
3027 any 3<sup>rd</sup> party defined Resources (see 7.8.4.4), and does not need to include Resources in the  
3028 IDD due to other clauses in this clause, then the IDD shall be an empty OpenAPI 2.0 file. An  
3029 example of an empty OpenAPI 2.0 file can be found in found in Annex **B.2**.
- 3030 – All other Resources that are individually addressable by a Client (i.e. the "href" can be resolved  
3031 and at least one operation is supported with a success path response) shall be listed in the IDD.
- 3032 – Per Resource the IDD shall include:
  - 3033 – All implemented methods
    - 3034 – For an OCF defined Resource Type, only the methods that are listed in the OpenAPI 2.0  
3035 definition are allowed to exist in the IDD. For an OCF defined Resource Type, methods  
3036 not listed in the OpenAPI 2.0 definition shall not exist in the IDD. The supported methods  
3037 contained in the IDD shall comply with the listed OCF Interfaces. For example, if the  
3038 POST method is listed in the IDD, then an OCF Interface that allows UPDATE will be  
3039 listed in the IDD.
  - 3040 – Per supported method:
    - 3041 – Implemented query parameters per method.
    - 3042 – This includes the supported OCF Interfaces ("if") as enum values.
    - 3043 – Schemas of the payload for the request and response bodies of the method.
    - 3044 – Where the schema provides the representation of a batch request or response ("oic.if.b")  
3045 the schema shall contain the representations for all Resource Types that may be  
3046 included within the batch representation. The representations shall be provided within  
3047 the IDD itself.
    - 3048 – The schema data shall be conveyed by the OpenAPI 2.0 schema.
    - 3049 – The OpenAPI 2.0 schema object shall comply with:
      - 3050 – The schemas shall be fully resolved, e.g. no references shall exist outside the  
3051 OpenAPI 2.0 file.
      - 3052 – The schemas shall list which OCF Interfaces are supported on the method.

- 3053           – The schemas shall list if a Property is optional or required.
- 3054           – The schemas shall include all Property validation keywords. Where an enum is
- 3055           defined the enum shall contain the values supported by the Device. When vendor
- 3056           defined extensions exist to the enum (defined in accordance to 7.8.4.4) these shall
- 3057           be included in the enum.
- 3058           – The schemas shall indicate if an Property is read only or read-write.
- 3059           – By means of the readOnly schema tag belonging to the Property.
- 3060           – Default value of readOnly is false as defined by OpenAPI 2.0.
- 3061           – The default value of the "rt" Property shall be used to indicate the supported
- 3062           Resource Types.
- 3063           – oneOf and anyOf constructs are allowed to be used as part of a OpenAPI 2.0 schema
- 3064           object. The OpenAPI 2.0 schema with oneOf and anyOf constructs can be found in
- 3065           Annex B.1.
- 3066   – For Atomic Measurements (see clause 7.8.4), the following apply:
- 3067           – The "rts" Property Value in the IDD shall include only the Resource Types the instance
- 3068           contains and not the theoretical maximal set allowed by the schema definition.
- 3069           – The Resources that are part of an Atomic Measurement, excluding the Atomic Measurement
- 3070           Resource itself, shall not be added to their own individual path in the IDD, as they are not
- 3071           individually addressable; however, the schemas for the composed Resource Types shall be
- 3072           provided in the IDD as part of the batch response definition along with the "href" for the
- 3073           Resource.
- 3074   Dynamic Resources (e.g. Resources that can be created on a request by a Client) shall have a
- 3075   URL definition which contains a URL identifier (e.g. using the {} syntax). A URL with {} identifies
- 3076   that the Resource definition applies to the whole group of Resources that may be created. The
- 3077   actual path may contain the Collection node that links to the Resource.

3078   Example of a URL with identifiers:

3079   /SceneListResURI/{SceneCollectionResURI}/{SceneMemberResURI}:

3080   When different Resource Types are allowed to be created in a Collection, then the different

3081   schemas for the CREATE method shall define all possible Resource Types that may be created.

3082   The schema construct oneOf allows the definition of a schema with selectable Resources. The

3083   oneOf construct allows the integration of all schemas and that only one existing sub schema shall

3084   be used to indicate the definition of the Resource that may be created.

3085   Example usage of oneOf JSON schema construct is shown in Figure 12:

```

3086  {
3087    "oneOf": [
3088      { <<subschema 1 definition>> },
3089      { << sub schema 2 definition >> }
3090    ...
3091  ]
3092  }
```

3093           **Figure 12 – Example usage of oneOf JSON schema**

3094   A Client using the IDD of a Device should check the version of the supported IDD of the Device.

3095   The OpenAPI 2.0 version is indicated in each file with the tag "swagger". Example of the 2.0

3096   supported version of the tag is: "swagger": "2.0". Later versions of this document may reference

3097   newer versions of the OpenAPI specification, for example 3.0.

A Device shall support one Resource with a Resource Type of "oic.wk.introspection" as defined in Table 27. The Resource with a Resource Type of "oic.wk.introspection" shall be included in the Resource "/oic/res".

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

**Table 27 – 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

Table 28 defines "oic.wk.introspection" Resource Type.

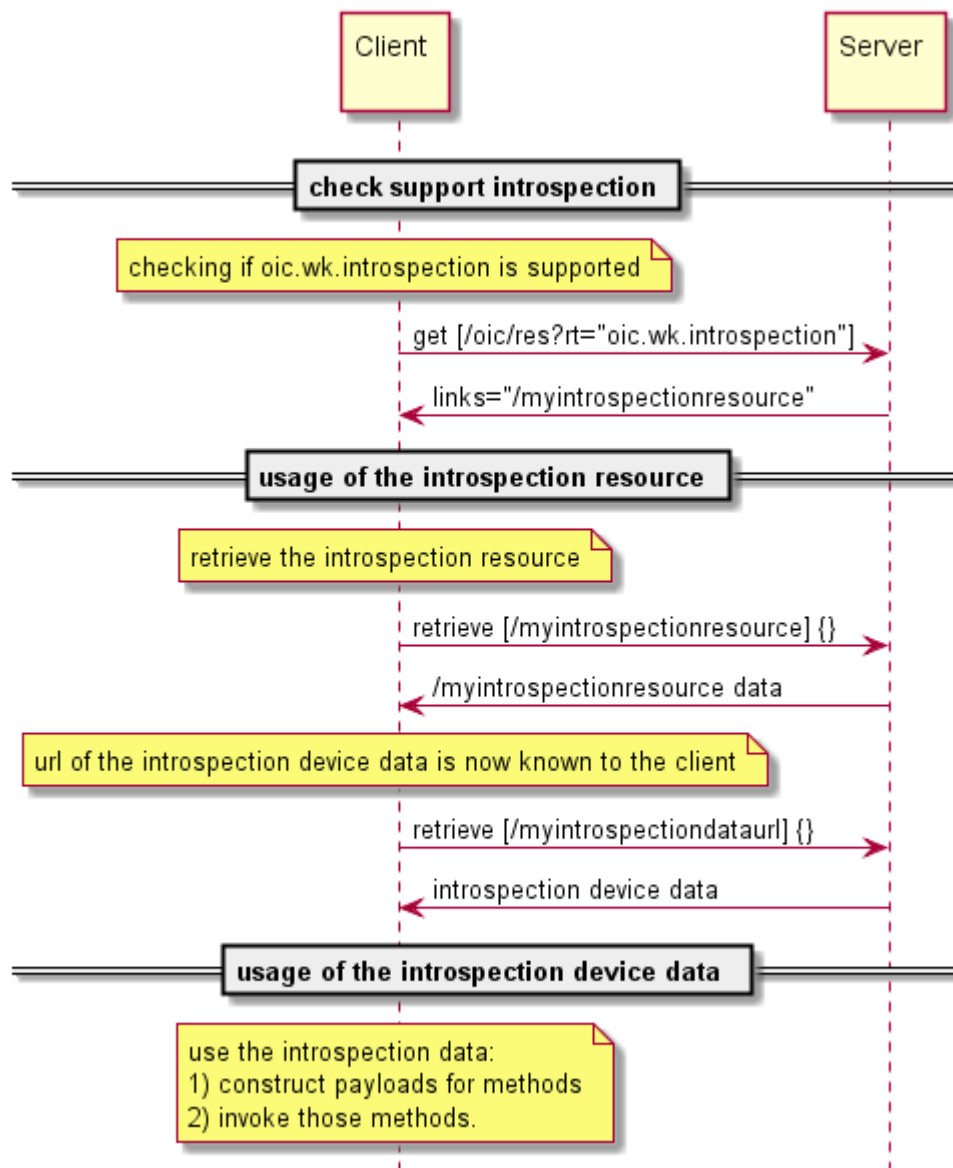
**Table 28 – "oic.wk.introspection" Resource Type definition**

Property title	Property name	Value type	Value rule	Unit	Access mode	Mandatory	Description
<b>urlInfo</b>	"urlInfo"	"array"	N/A	N/A	R	Yes	array of objects
<b>url</b>	"url"	"string"	"uri"	N/A	R	Yes	URL to the hosted payload
<b>protocol</b>	"protocol"	"string"	"enum"	N/A	R	Yes	Protocol definition to retrieve the Introspection Device Data from the url.
<b>content-type</b>	"content-type"	"string"	"enum"	N/A	R	No	content type of the url.
<b>version</b>	"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.

#### 11.4.2 Usage of Introspection

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

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



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

## 11.5 Semantic Tags

### 11.5.1 Introduction

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

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



**11.5.2 Semantic Tag definitions**

**11.5.2.1 Relative and descriptive position Semantic Tags**

**11.5.2.1.1 Introduction**

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

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

The "tag-pos-desc" Semantic Tag as defined in Table 29 describes the position of the Resource as a descriptive position. If the tag is not exposed it conveys the same meaning as if the tag is exposed with a value of "unknown". The value for the "tag-pos-desc" Semantic Tag if exposed, shall be a string containing a value from the enumeration detailed in Annex C. The population of the Semantic Tag is defined by the Device vendor and shall not be mutable by a Client.

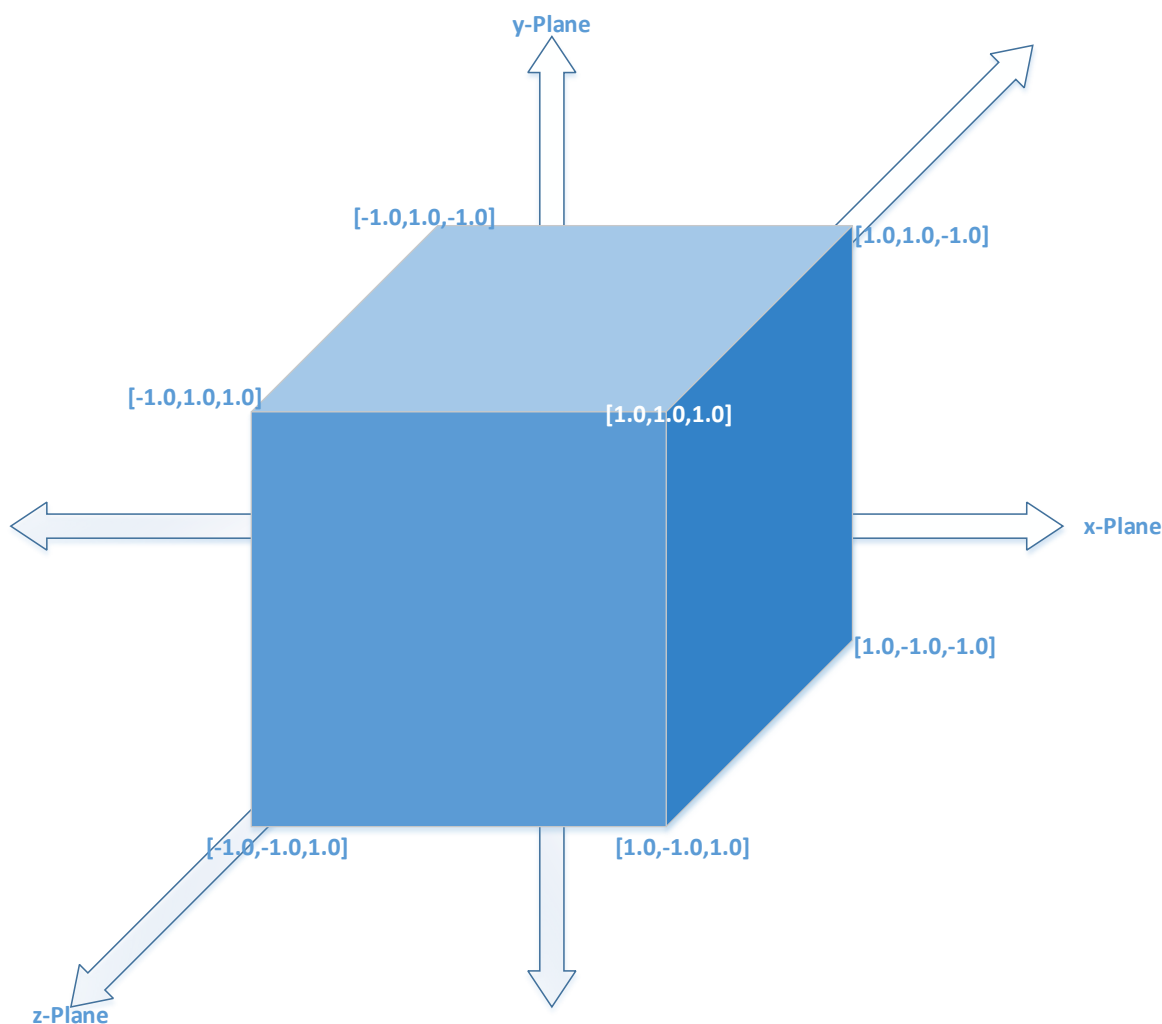
**Table 29 – "tag-pos-desc" Semantic Tag definition**

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

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

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

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



**Figure 14 – "tag-pos-rel" definition**

The "tag-pos-rel" Semantic Tag value is defined by the Device vendor and shall not be mutable by a Client. This is detailed in Table 30.

**Table 30 – "tag-pos-rel" Semantic Tag definition**

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

## 11.5.2.2 Functional behaviour Semantic Tags

### 11.5.2.2.1 Introduction

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

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

#### 11.5.2.2.2 "tag-func-desc" or function description Semantic Tag

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

This "tag-func-desc" Semantic Tag is detailed in Table 31.

**Table 31 – "tag-func-desc" Semantic Tag definition**

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

#### 11.5.2.3 Location Semantic Tags

##### 11.5.2.3.1 Introduction

Consider a Bridge, Resource Directory or other similar concept whereby the Link to the Device Resource ("oic.wk.d") that is exposed may reference or relate to a physically separate Device. In such a case the ability to annotate the Link to the Device Resource with location information becomes useful. Additionally, in a deployment of multiple similar or identical Devices, the ability to annotate the Device with where it is deployed assists in disambiguation.

##### 11.5.2.3.2 "tag-locn" or location description Semantic Tag

The "tag-locn" Semantic Tag may be exposed as a Link Parameter for the Device Resource, it describes the physical location of the target Device, it shall not be exposed as a Link Parameter for any other Resource Type. If the tag is not exposed it conveys the same meaning as if the tag is exposed with a value of "unknown". The initial value for the "tag-locn" Semantic Tag if exposed shall be "unknown". This Link Parameter shall not contain any 3<sup>rd</sup> party defined values.

The "tag-locn" shall be exposed as string containing a value from the enumeration ("locn-descriptions") defined in Annex C. The tag is detailed in Table 32.

An instance of "tag-locn" may be updated by a Client by modifying the reflected instance of this value that is present in the Configuration Resource, see [1].

**Table 32 – "tag-locn" Semantic Tag definition**

Semantic Tag Name	Type	Contents	Value example
tag-locn	Enumeration	See Annex C	"tag-locn": "familyroom"

## 12 Messaging

### 12.1 Introduction

This clause specifies the protocol messaging mapping to the CRUDN messaging operations (clause 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols is expected in later version of this document. All the Property information from the Resource model shall be carried within the message payload. This payload shall be generated in the Resource model layer

3199 and shall be encapsulated in the data connectivity layer. The message header shall only be used  
 3200 to describe the message payload (e.g., verb, mime-type, message payload format), in addition to  
 3201 the mandatory header fields defined in a messaging protocol (e.g., CoAP) specification. If the  
 3202 message header does not support this, then this information shall also be carried in the message  
 3203 payload. Resource model information shall not be included in the message header structure unless  
 3204 the message header field is mandatory in the messaging protocol specification.

3205 When a Resource is specified with a RESTful description language like OpenAPI 2.0 then the HTTP  
 3206 syntax definitions are used in the description (e.g., HTTP syntax for the CRUDN operations, status  
 3207 codes, etc). The HTTP syntax will be mapped to the actual used web transfer protocol (e.g., CoAP).

3208 The communication is largely based on UDP and UDP has defined the Maximum Transmission Unit  
 3209 (MTU). All UDP payload size communications shall not exceed the MTU size as per by the  
 3210 IETF RFC 8085 clause 3.2. This is to avoid being dependent on package reassembly by the  
 3211 operating systems.

## 3212 **12.2 Mapping of CRUDN to CoAP**

### 3213 **12.2.1 Overview**

3214 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in clause  
 3215 12.2.3. A Device implementing CoAP shall conform to IETF RFC 7641 to implement the CoAP  
 3216 Observe option. Support for CoAP block transfer when the payload is larger than the MTU is defined  
 3217 in 12.2.8.

### 3218 **12.2.2 URIs**

3219 An OCF: URI is mapped to a coap: URI by replacing the scheme name "ocf" with "coap" if unsecure  
 3220 or "coaps" if secure before sending over the network by the requestor. Similarly on the receiver  
 3221 side, the scheme name is replaced with "ocf".

3222 Any query string that is present within the URI is encoded as one or more URI-Query Options as  
 3223 defined in IETF RFC 7252 clause 6.4.

### 3224 **12.2.3 CoAP method with request and response**

#### 3225 **12.2.3.1 Overview**

3226 Every request has a CoAP method that realizes the request. The primary methods and their  
 3227 meanings are shown in Table 33, which provides the mapping of GET/POST/DELETE methods to  
 3228 CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides the generic  
 3229 behaviours when using these methods, however Resource OCF Interfaces may modify these  
 3230 generic semantics. The HTTP codes in the RESTful descriptions will be translated as described in  
 3231 IETF RFC 8075 clause 7 Response Code Mapping. CoAP methods not listed in Table 33 are not  
 3232 supported.

3233 **Table 33 – CoAP request and response**

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

<b>POST for UPDATE</b>	<ul style="list-style-type: none"> <li>- <b>Method code:</b> POST (0.02).</li> <li>- <b>Request URI:</b> an existing URI for the Resource to be updated.</li> <li>- <b>Payload:</b> representation of the Resource to be updated.</li> </ul>	- <b>Response Code:</b> success (2.xx) or error (4.xx or 5.xx).
<b>DELETE for DELETE</b>	<ul style="list-style-type: none"> <li>- <b>Method code:</b> DELETE (0.04).</li> <li>- <b>Request URI:</b> an existing URI for the Resource to be deleted.</li> </ul>	- <b>Response code:</b> success (2.xx) or error (4.xx or 5.xx).

3234

3235

### 3236 12.2.3.2 CREATE with POST

3237 POST with the "oic.if.create" OCF Interface query parameter (i.e., "POST ?if=oic.if.create") shall  
 3238 be used only in situations where the request URI is valid, that is it is the URI of an existing Resource  
 3239 on the Server that is processing the request. If no such Resource is present, the Server shall  
 3240 respond with an error response code of 4.xx. The use of POST for CREATE shall use an existing  
 3241 request URI which identifies the Resource on the Server responsible for creation. The URI of the  
 3242 created Resource is determined by the Server and provided to the Client in the response.

3243 A Client shall include the representation of the new Resource in the request payload. The new  
 3244 resource representation in the payload shall have all the necessary Properties to create a valid  
 3245 Resource instance, i.e. the created Resource should be able to properly respond to the valid  
 3246 Request with mandatory OCF Interface (e.g., "GET with ?if=oic.if.baseline").

3247 Upon receiving the POST request, the Server shall either:

- 3248 – Create the new Resource with a new URI, respond with the new URI for the newly created  
 3249 Resource and a success response code (2.xx); or
- 3250 – respond with an error response code (4.xx or 5.xx).

### 3251 12.2.3.3 RETRIEVE with GET

3252 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of  
 3253 the target Resource identified by the request URI.

3254 Upon receiving the GET request, the Server shall either:

- 3255 – Send back the response with the representation of the target Resource with a success response  
 3256 code (2.xx); or
- 3257 – respond with an error response code (4.xx or 5.xx) or ignore it (e.g. non-applicable multicast  
 3258 GET).

3259 GET is a safe method and is idempotent.

### 3260 12.2.3.4 UPDATE with POST

3261 POST shall be used only in situations where the request URI is valid, that is it is the URI of an  
 3262 existing Resource on the Server that is processing the request. If no such Resource is present, the  
 3263 Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE  
 3264 Property values of an existing Resource.

3265 Upon receiving the request, the Server shall either:

- 3266 – Apply the request to the Resource identified by the request URI in accordance with the applied  
 3267 OCF Interface (i.e. POST for non-existent Properties is ignored) and send back a response with  
 3268 a success response code (2.xx); or

3269 – respond with an error response code (4.xx or 5.xx). Note that if the representation in the payload  
 3270 is incompatible with the target Resource for POST using the applied OCF Interface (i.e. the  
 3271 overwrite semantic cannot be honored because of read-only Property in the payload), then the  
 3272 error response code 4.xx shall be returned.

### 3273 **12.2.3.5 DELETE with DELETE**

3274 DELETE shall be used for DELETE operation. The DELETE method requests that the Resource  
 3275 identified by the request URI be deleted.

3276 Upon receiving the DELETE request, the Server shall either:

- 3277 – Delete the target Resource and send back a response with a success response code (2.xx); or
- 3278 – respond with an error response code (4.xx or 5.xx).

3279 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

### 3280 **12.2.4 Content-Format negotiation**

3281 The Framework mandates support of CBOR, however it allows for negotiation of the payload body  
 3282 if more than one Content-Format (e.g. CBOR and JSON) is supported by an implementation. In this  
 3283 case the Accept Option defined in clause 5.10.4 of IETF RFC 7252 shall be used to indicate which  
 3284 Content-Format (e.g. JSON) is requested by the Client.

3285 The Content-Formats supported are shown in Table 34.

3286 **Table 34 – OCF Content-Formats**

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

3287  
 3288 Clients shall include a Content-Format Option in every message that contains a payload. Servers  
 3289 shall include a Content-Format Option for all success (2.xx) responses with a payload body. Per  
 3290 IETF RFC 7252 clause 5.5.1, Servers shall include a Content-Format Option for all error (4.xx or  
 3291 5.xx) responses with a payload body unless they include a Diagnostic Payload; error responses  
 3292 with a Diagnostic Payload do not include a Content-Format Option. The Content-Format Option  
 3293 shall use the ID column numeric value from Table 34. An OCF vertical may mandate a specific  
 3294 Content-Format Option.

3295 Clients shall also include an Accept Option in every request message. The Accept Option shall  
 3296 indicate the required Content-Format as defined in Table 34 for response messages. The Server  
 3297 shall return the required Content-Format if available. If the required Content-Format cannot be  
 3298 returned, then the Server shall respond with an appropriate error message.

### 3299 **12.2.5 OCF-Content-Format-Version information**

3300 Servers and Clients shall include the OCF-Content-Format-Version Option in both request and  
 3301 response messages with a payload. Clients shall include the OCF-Accept-Content-Format-Version  
 3302 Option in request messages. The OCF-Content-Format-Version Option and OCF-Accept-Content-  
 3303 Format-Version Option are specified as Option Numbers in the CoAP header as shown in Table 35.

**Table 35 – OCF-Content-Format-Version and OCF-Accept-Content-Format-Version Option Numbers**

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

The value of both the OCF-Accept-Content-Format-Version Option and the OCF-Content-Format-Version Option is a two-byte unsigned integer that is used to define the major, minor and sub versions. The major and minor versions are represented by 5 bits and the sub version is represented by 6 bits as shown in Table 36.

**Table 36 – 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

Table 37 illustrates several examples:

**Table 37 – 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

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

### 12.2.6 Content-Format policy

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

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

- All Client Devices shall support OCF-Content-Format-Version Option set to "1.0.0" and higher.
- All Client Devices shall support OCF-Accept-Content-Format-Version Option set to "1.0.0" and higher.
- A Client shall send a discovery request message with its Accept Option set to "application/vnd.ocf+cbor", and its OCF-Accept-Content-Format-Version Option matching its highest supported version.
- A Server shall respond to a Client's discovery request that is higher than its OCF-Content-Format-Version by responding with its Content-Format Option set to "application/vnd.ocf+cbor", and OCF-Content-Format-Version matching its highest supported version. The response

3333 representation shall be encoded with the OCF-Content-Format-Version matching the Server's  
3334 highest supported version.

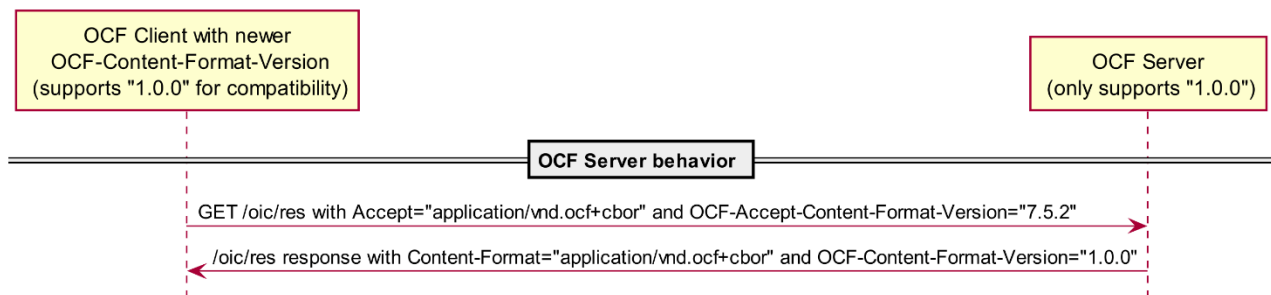
3335 – A Server may support previous Content-Formats and OCF-Content-Format-Versions to support  
3336 backward compatibility with previous versions.

3337 – For a Server that supports multiple OCF-Content-Format-Version Options, the Server should  
3338 attempt to respond with an OCF-Content-Format-Version that matches the OCF-Accept-  
3339 Content-Format-Version of the request.

3340 To maintain compatibility between Devices implemented to different versions of this document,  
3341 Devices should follow the policy as described in Figure 15.

3342 The OCF Clients in Figure 15 support sending Content-Format Option set to  
3343 "application/vnd.ocf+cbor", Accept Option set to "application/vnd.ocf+cbor", OCF-Content-Format-  
3344 Version Option set to "1.0.0", and OCF-Accept-Content-Format-Version Option set to "1.0.0"  
3345 (representing OCF 1.0 and later Clients). The OCF Servers in Figure 15 support sending Content-  
3346 Format Option set to "application/vnd.ocf+cbor" and OCF-Content-Format-Version Option set to  
3347 "1.0.0" (representing OCF 1.0 and later Servers).

3348



3349

3350 **Figure 15 – Content-Format Policy for backward compatible OCF Clients negotiating lower**  
3351 **OCF Content-Format-Version**

### 3352 12.2.7 CRUDN to CoAP response codes

3353 The mapping of CRUDN operations response codes to CoAP response codes are identical to the  
3354 response codes defined in IETF RFC 7252.

### 3355 12.2.8 CoAP block transfer

3356 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT  
3357 devices. However scenarios can be envisioned in which an application needs to transfer larger  
3358 payloads.

3359 CoAP block-wise transfer as defined in IETF RFC 7959 shall be used by all Servers which generate  
3360 a content payload that would exceed the size of a CoAP datagram as the result of handling any  
3361 defined CRUDN operation.

3362 Similarly, CoAP block-wise transfer as defined in IETF RFC 7959 shall be supported by all Clients.  
3363 The use of block-wise transfer is applied to both the reception of payloads as well as transmission  
3364 of payloads that would exceed the size of a CoAP datagram.

3365 A Client may support both the block1 (as descriptive) and block2 (as control) options as described  
3366 by IETF RFC 7959. A Server may support both the block1 (as control) and block2 (as descriptive)  
3367 options as described by IETF RFC 7959.



### 12.2.9 Generic requirements for CoAP multicast

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

- Devices shall join the All OCF Nodes multicast groups (as defined in [IANA IPv6 Multicast Address Space Registry]) with scopes 2, 3, and 5 (i.e., ff02::158, ff03::158 and ff05::158) and shall listen on the port 5683. For compliance to IETF RFC 7252 a Device may additionally join the All CoAP Nodes multicast groups.
- Clients intending to discover Resources shall join the multicast groups as defined in the first bullet.
- Clients shall send multicast requests to the All OCF Nodes multicast group address with scope 2 ("ff02::158") or with scope 5 ("ff05::158") at port "5683". The requested URI shall be the fixed local path of the target Resource optionally followed by query parameters. For compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast groups.
- To discover Devices on a low-rate wireless personal area network (LR-WPAN) [see IETF RFC 7346], Clients should send additional discovery requests (GET request) to the All OCF Nodes multicast group address with REALM\_LOCAL scope 3 ("ff03::158") at port "5683". The set of replying Devices then can be used to distinguish if the Device is SITE\_LOCAL or REALM\_LOCAL to the Client discovering the Devices. Such request shall use the IPv6 hop limit with a value of 255. If the Client sends discovery requests to All OCF Nodes, then for compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast groups with the same REALM\_LOCAL scope with the IPv6 hop limit value of 255.
- Clients should send discovery requests (GET request) to the All OCF Nodes multicast group address with SITE\_LOCAL scope 5 ("ff05::158") at port "5683". Such request shall use the IPv6 hop limit with a value of 255. If the Client sends discovery requests to All OCF Nodes, then for compliance to IETF RFC 7252 a Client may additionally send to the All CoAP Nodes multicast groups with the same SITE\_LOCAL scope with the IPv6 hop limit value of 255.
- The multicast request shall be permitted by matching the request to an ACE which permits unauthenticated access to the target Resource as described in ISO/IEC 30118-2.
- Handling of multicast requests shall be as described in clause 8 of IETF RFC 7252 and clause 4.1 in IETF RFC 6690.
- Devices which receive the request shall respond, subject to query parameter processing specific to the requested Resource.

### 12.2.10 Setting timeout on response to a confirmable request

The timeout specified by "oic.wk.res:eps[:lat]", when present, should only be taken into account by the Client when the Server is in the "ready for normal operation state" [see clause 8.5 in ISO/IEC 30118-2] and the request made is a confirmable request. The Server should only enable the state that will cause latency when in "ready for normal operation state" [see clause 8.5 in ISO/IEC 30118-2]. In all other states the Server should respond with timeouts as identified in IETF RFC 7252.

### 12.2.11 Mapping the error response payload

The error response payload as defined in clause 7.10 shall be included as a diagnostic payload as described in IETF RFC 7252 clause 5.5.2. The diagnostic payload shall be encoded in ASCII.

## 12.3 Mapping of CRUDN to CoAP serialization over TCP

### 12.3.1 Overview

In environments where TCP is already available, CoAP can take advantage of it to provide reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For example, consider a cloud application acting as a Client and the Server is located at the user's home. A Server which already support CoAP as a messaging protocol could easily support CoAP

3416 serialization over TCP rather than utilizing another messaging protocol. A Device implementing  
3417 CoAP Serialization over TCP shall conform to IETF RFC 8323.

### 3418 **12.3.2 URIs**

3419 When UDP is blocked, Clients are dependent on pre-configured details of the Device to determine  
3420 if the Device supports CoAP serialization over TCP. When UDP is not-blocked, a Device which  
3421 supports CoAP serialization over TCP shall populate the "eps" Parameter in the "/oic/res" response,  
3422 as defined in 10.2, with the URI scheme(s) as defined in clause 8.1 or 8.2 of IETF RFC 8323. For  
3423 the "coaps+tcp" URI scheme, as defined in clause 8.2 of IETF RFC 8323, IETF RFC 7301 shall be  
3424 used. In addition, the URIs used for CoAP serialization over TCP shall conform to 12.2.2 by  
3425 substituting the scheme names with the scheme names defined in clauses 8.1 and 8.2 of  
3426 IETF RFC 8323 respectively.

### 3427 **12.3.3 CoAP method with request and response**

3428 The CoAP methods used for CoAP serialization over TCP shall conform to 12.2.3.

### 3429 **12.3.4 Content-Format negotiation**

3430 The Content Format negotiation used for CoAP serialization over TCP shall conform to 12.2.4.

### 3431 **12.3.5 OCF-Content-Format-Version information**

3432 The OCF Content Format Version information used for CoAP serialization over TCP shall conform  
3433 to 12.2.5.

### 3434 **12.3.6 Content-Format policy**

3435 The Content Format policy used for CoAP serialization over TCP shall conform to 12.2.6.

### 3436 **12.3.7 CRUDN to CoAP response codes**

3437 The CRUDN to CoAP response codes for CoAP serialization over TCP shall conform to 12.2.7.

### 3438 **12.3.8 CoAP block transfer**

3439 The CoAP block transfer for CoAP serialization over TCP shall conform to clause 6 of  
3440 IETF RFC 8323.

### 3441 **12.3.9 Keep alive (connection health)**

3442 The Device that initiated the CoAP over TCP connection shall send a Ping message as described  
3443 in clause 5.4 in IETF RFC 8323. The Device to which the connection was made may send a Ping  
3444 message. The recipient of any Ping message shall send a Pong message as described in clause  
3445 5.4 in IETF RFC 8323.

3446 Both sides of an established CoAP over TCP connection may send subsequent Ping (and  
3447 corresponding Pong) messages.

### 3448 **12.3.10 CoAP using a proxy**

3449 In cases that a request is made to a forwarding proxy, the option proxy-uri (clause 5.10.2 of  
3450 IETF RFC 7252) shall be used. The format of the information in the proxy-uri option includes the  
3451 OCF Device information. The proxy-uri shall have the format of an OCF URI as described in clause  
3452 6.2.2. The authority will have the same value as "oic.wk.d:uuid" of the targeted Device.

### 3453 **12.3.11 Mapping the error response payload**

3454 The mapping of the error response payload for CoAP serialization over TCP shall conform to clause  
3455 12.2.11.

#### 3456 **12.4 Payload Encoding in CBOR**

3457 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to  
3458 JSON from CBOR in accordance with IETF RFC 7049 clause 4 unless otherwise specified in this  
3459 clause.

3460 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types  
3461 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-  
3462 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation  
3463 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer  
3464 numbers shall be within the closed interval  $[-2^{53}, 2^{53}]$ . Properties defined as a JSON number  
3465 should be encoded as integers whenever possible; if this is not possible Properties defined as a  
3466 JSON number should use single-precision if the loss of precision does not affect the quality of  
3467 service, otherwise the Property shall use double-precision.

3468 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values  
3469 in any position. If a Property defined as a JSON integer is received encoded other than as an  
3470 integer, the implementation may reject this encoding using a final response as appropriate for the  
3471 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a Property is  
3472 defined as a JSON number an implementation shall accept integers, single- and double-precision  
3473 floating point.

#### 3474 **13 Security**

3475 The details for handling security and privacy are specified in ISO/IEC 30118-2.

## Annex A (normative)

### Resource Type definitions

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

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

**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

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

##### A.2.1 Introduction

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

##### A.2.2 Example URI

/AtomicMeasurementResURI

##### A.2.3 Resource type

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

##### A.2.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Atomic Measurement links list representation",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2018-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/AtomicMeasurementResURI?if=oic.if.ll": {
      "get": {
        "description": "The oic.if.ll OCF Interface exposes a representation
of the Links",
```

```

3516         "parameters": [
3517     {
3518         "$ref": "#/parameters/interface-all"
3519     }
3520 ],
3521     "responses": {
3522         "200": {
3523             "description": "",
3524             "x-example": [{
3525                 "href": "/temperature",
3526                 "rt": ["oic.r.temperature"],
3527                 "if": ["oic.if.s", "oic.if.baseline"]
3528             }],
3529             {
3530                 "href": "/bodylocation",
3531                 "rt": ["oic.r.body.location.temperature"],
3532                 "if": ["oic.if.s", "oic.if.baseline"]
3533             },
3534             {
3535                 "href": "/timestamp",
3536                 "rt": ["oic.r.time.stamp"],
3537                 "if": ["oic.if.s", "oic.if.baseline"]
3538             }],
3539             "schema": {
3540                 "$ref": "#/definitions/links"
3541             }
3542         }
3543     }
3544 },
3545     "/AtomicMeasurementResURI?if=oic.if.b": {
3546         "get": {
3547             "description": "The oic.if.b OCF Interface returns data items
3548 retrieved from Resources pointed to by the Links.\n",
3549             "parameters": [
3550     {
3551         "$ref": "#/parameters/interface-all"
3552     }
3553 ],
3554     "responses": {
3555         "200": {
3556             "description": "Normal response, no errors, all
3557 Properties are returned correctly\n",
3558             "x-example": [{
3559                 "href": "/temperature",
3560                 "rep": {
3561                     "temperature": 38,
3562                     "units": "C",
3563                     "range": [25, 45]
3564                 }
3565             }],
3566             {
3567                 "href": "/bodylocation",
3568                 "rep": {
3569                     "bloc": "ear"
3570                 }
3571             },
3572             {
3573                 "href": "/timestamp",
3574                 "rep": {
3575                     "timestamp": "2007-04-05T14:30+09:00"
3576                 }
3577             }],
3578             "schema": {
3579                 "$ref": "#/definitions/batch-retrieve"
3580             }
3581         }
3582     }
3583 },
3584     "/AtomicMeasurementResURI?if=oic.if.baseline": {

```

```

3587         "get": {
3588             "description": "The oic.if.baseline OCF Interface exposes a
3589 representation of the links and\nthe Common Properties of the Atomic Measurement Resource.\n",
3590             "parameters": [
3591                 {
3592                     "$ref": "#/parameters/interface-all"
3593                 }
3594             ],
3595             "responses": {
3596                 "200": {
3597                     "description": "",
3598                     "x-example": {
3599                         "rt": ["oic.wk.atomicmeasurement"],
3600                         "if": ["oic.if.b", "oic.if.ll",
3601                             "oic.if.baseline"],
3602                         "rts": ["oic.r.temperature",
3603 "oic.r.body.location.temperature", "oic.r.time.stamp"],
3604                         "rts-m": ["oic.r.temperature",
3605 "oic.r.body.location.temperature", "oic.r.time.stamp"],
3606                         "links": [{
3607                             "href": "/temperature",
3608                             "rt": ["oic.r.temperature"],
3609                             "if": ["oic.if.s", "oic.if.baseline"]
3610                         },
3611                         {
3612                             "href": "/bodylocation",
3613                             "rt":
3614 ["oic.r.body.location.temperature"],
3615                             "if": ["oic.if.s", "oic.if.baseline"]
3616                         },
3617                         {
3618                             "href": "/timestamp",
3619                             "rt": ["oic.r.time.stamp"],
3620                             "if": ["oic.if.s", "oic.if.baseline"]
3621                         }
3622                     ],
3623                     "schema": {
3624                         "$ref": "#/definitions/baseline"
3625                     }
3626                 }
3627             }
3628         }
3629     },
3630     "parameters": {
3631         "interface-all": {
3632             "in": "query",
3633             "name": "if",
3634             "type": "string",
3635             "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"]
3636         }
3637     },
3638     "definitions": {
3639         "links": {
3640             "type": "array",
3641             "items": {
3642                 "$ref": "#/definitions/oic.oic-link"
3643             }
3644         },
3645         "batch-retrieve": {
3646             "title": "Collection Batch Retrieve Format (auto merged)",
3647             "minItems": 1,
3648             "items": {
3649                 "additionalProperties": true,
3650                 "properties": {
3651                     "href": {
3652                         "$ref":
3653 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3654 schema.json#/definitions/href"
3655                     },
3656                     "rep": {

```

```

3658         "oneOf": [{
3659             "description": "The response payload from a
3660 single Resource",
3661             "type": "object"
3662         },
3663         {
3664             "description": " The response payload from a
3665 Collection (batch) Resource",
3666             "items": {
3667                 "properties": {
3668                     "anchor": {
3669                         "$ref":
3670 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3671 schema.json#/definitions/anchor"
3672                     },
3673                     "di": {
3674                         "$ref":
3675 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3676 schema.json#/definitions/di"
3677                     },
3678                     "eps": {
3679                         "$ref":
3680 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3681 schema.json#/definitions/eps"
3682                     },
3683                     "href": {
3684                         "$ref":
3685 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3686 schema.json#/definitions/href"
3687                     },
3688                     "if": {
3689                         "description": "The OCF
3690 Interface set supported by this Resource",
3691                         "items": {
3692                             "enum": [
3693                                 "oic.if.baseline",
3694                                 "oic.if.ll",
3695                                 "oic.if.b",
3696                                 "oic.if.rw",
3697                                 "oic.if.r",
3698                                 "oic.if.a",
3699                                 "oic.if.s"],
3700                             "type":
3701 "string"
3702                         },
3703                         "minItems": 1,
3704                         "uniqueItems": true,
3705                         "type": "array"
3706                     },
3707                     "ins": {
3708                         "$ref":
3709 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3710 schema.json#/definitions/ins"
3711                     },
3712                     "p": {
3713                         "$ref":
3714 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3715 schema.json#/definitions/p"
3716                     },
3717                     "rel": {
3718                         "description": "The relation of the target URI
3719 referenced by the Link to the context URI",
3720                         "oneOf": [
3721                             {
3722                                 "$ref":
3723 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3724 schema.json#/definitions/rel_array"
3725                             },
3726                             {
3727                                 "$ref":
3728

```

```

3729 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3730 schema.json#/definitions/rel_string"
3731     }
3732   ],
3733 },
3734     "rt": {
3735       "description":
3736         "Resource Type of the Resource",
3737       "items": {
3738         "maxLength":
3739           64,
3740         "type":
3741           "string"
3742       },
3743       "minItems": 1,
3744       "uniqueItems": true,
3745       "type": "array"
3746     },
3747     "title": {
3748       "$ref":
3749         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3750         schema.json#/definitions/title"
3751     },
3752     "type": {
3753       "$ref":
3754         "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3755         schema.json#/definitions/type"
3756     }
3757   },
3758   "required": [
3759     "href",
3760     "rt",
3761     "if"
3762   ],
3763   "type": "object"
3764 },
3765 "type": "array"
3766 ]]
3767   }
3768 },
3769 "required": [
3770   "href",
3771   "rep"
3772 ],
3773 "type": "object"
3774 },
3775 "type": "array"
3776 },
3777 "baseline": {
3778   "properties": {
3779     "links": {
3780       "description": "A set of simple or individual Links.",
3781       "items": {
3782         "$ref": "#/definitions/oic.oic-link"
3783       },
3784       "type": "array"
3785     },
3786     "n": { "$ref":
3787       "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3788       schema.json#/definitions/n"},
3789     "id": { "$ref":
3790       "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
3791       schema.json#/definitions/id"},
3792     "rt": {
3793       "description": "Resource Type of this Resource",
3794       "items": {
3795         "enum": ["oic.wk.atomicmeasurement"],
3796         "type": "string",
3797         "maxLength": 64
3798       },
3799       "minItems": 1,

```



```

3800         "readOnly": true,
3801         "uniqueItems": true,
3802         "type": "array"
3803     },
3804     "rts": {
3805         "description": "An array of Resource Types that are supported
3806 within an array of Links exposed by the Resource",
3807         "items": {
3808             "maxLength": 64,
3809             "type": "string"
3810         },
3811         "minItems": 1,
3812         "readOnly": true,
3813         "uniqueItems": true,
3814         "type": "array"
3815     },
3816     "rts-m": {
3817         "description": "An array of Resource Types that are mandatory
3818 to be exposed within an array of Links exposed by the Resource",
3819         "items": {
3820             "maxLength": 64,
3821             "type": "string"
3822         },
3823         "minItems": 1,
3824         "readOnly": true,
3825         "uniqueItems": true,
3826         "type": "array"
3827     },
3828     "if": {
3829         "description": "The OCF Interface set supported by this
3830 Resource",
3831         "items": {
3832             "enum": ["oic.if.b", "oic.if.ll", "oic.if.baseline"],
3833             "type": "string"
3834         },
3835         "minItems": 3,
3836         "readOnly": true,
3837         "uniqueItems": true,
3838         "type": "array"
3839     }
3840 },
3841 "type": "object",
3842 "required": [
3843     "rt",
3844     "if",
3845     "links"
3846 ],
3847 },
3848 "oic.oic-link": {
3849     "properties": {
3850         "anchor": {
3851             "$ref":
3852 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3853 schema.json#/definitions/anchor"
3854         },
3855         "di": {
3856             "$ref":
3857 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3858 schema.json#/definitions/di"
3859         },
3860         "eps": {
3861             "$ref":
3862 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3863 schema.json#/definitions/eps"
3864         },
3865         "href": {
3866             "$ref":
3867 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3868 schema.json#/definitions/href"
3869         },
3870         "if": {

```

```

3871         "description": "The OCF Interface set supported by this
3872 Resource",
3873         "items": {
3874             "enum": [
3875                 "oic.if.baseline",
3876                 "oic.if.ll",
3877                 "oic.if.b",
3878                 "oic.if.rw",
3879                 "oic.if.r",
3880                 "oic.if.a",
3881                 "oic.if.s"],
3882             "type": "string"
3883         },
3884         "minItems": 1,
3885         "uniqueItems": true,
3886         "type": "array"
3887     },
3888     "ins": {
3889         "$ref":
3890 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3891 schema.json#/definitions/ins"
3892     },
3893     "p": {
3894         "$ref":
3895 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3896 schema.json#/definitions/p"
3897     },
3898     "rel": {
3899         "description": "The relation of the target URI referenced by the Link to the context URI",
3900         "oneOf": [
3901             {
3902                 "$ref":
3903 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3904 schema.json#/definitions/rel_array"
3905             },
3906             {
3907                 "$ref":
3908 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3909 schema.json#/definitions/rel_string"
3910             }
3911         ]
3912     },
3913     "rt": {
3914         "description": "Resource Type of the Resource",
3915         "items": {
3916             "maxLength": 64,
3917             "type": "string"
3918         },
3919         "minItems": 1,
3920         "uniqueItems": true,
3921         "type": "array"
3922     },
3923     "title": {
3924         "$ref":
3925 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3926 schema.json#/definitions/title"
3927     },
3928     "type": {
3929         "$ref":
3930 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
3931 schema.json#/definitions/type"
3932     },
3933     "required": [
3934         "href",
3935         "rt",
3936         "if"
3937     ],
3938     "type": "object"
3939 }
3940 }
3941 }

```

3942 }  
3943

#### 3944 A.2.5 Property definition

3945 Table A.2 defines the Properties that are part of the "oic.wk.atomicmeasurement" Resource Type.

3946 **Table A.2 – The Property definitions of the Resource with type "rt" =**  
3947 **"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	

## A.2.6 CRUDN behaviour

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

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

Create	Read	Update	Delete	Notify
	get			observe

## A.3 Collection

### A.3.1 Introduction

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

### A.3.2 Example URI

/CollectionResURI

### A.3.3 Resource type

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

### A.3.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Collection",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/CollectionResURI?if=oic.if.ll" : {
      "get": {
        "description": "Collection Resource Type contains Properties and Links.\n\nThe oic.if.ll OCF
```

```

3989 Interface exposes a representation of the Links\n",
3990     "parameters": [
3991         {
3992             "$ref": "#/parameters/interface-all"
3993         }
3994     ],
3995     "responses": {
3996         "200": {
3997             "description": "",
3998             "x-example": [
3999                 {
4000                     "href": "/switch",
4001                     "rt": ["oic.r.switch.binary"],
4002                     "if": ["oic.if.a", "oic.if.baseline"],
4003                     "eps": [
4004                         {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
4005                         {"ep": "coaps://[fe80::b1d6]:1122"},
4006                         {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
4007                     ]
4008                 },
4009                 {
4010                     "href": "/airFlow",
4011                     "rt": ["oic.r.airflow"],
4012                     "if": ["oic.if.a", "oic.if.baseline"],
4013                     "eps": [
4014                         {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
4015                         {"ep": "coaps://[fe80::b1d6]:1122"},
4016                         {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
4017                     ]
4018                 }
4019             ],
4020             "schema": {
4021                 "$ref": "#/definitions/slinks"
4022             }
4023         }
4024     }
4025 },
4026 {
4027     "/CollectionResURI?if=oic.if.baseline" : {
4028         "get": {
4029             "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.baseline
4030 OCF Interface exposes a representation of\nthe Links and the Properties of the Collection Resource
4031 itself\n",
4032             "parameters": [
4033                 {
4034                     "$ref": "#/parameters/interface-all"
4035                 }
4036             ],
4037             "responses": {
4038                 "200": {
4039                     "description": "",
4040                     "x-example": {
4041                         "rt": ["oic.wk.col"],
4042                         "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
4043                         "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
4044                         "rts-m": [ "oic.r.switch.binary" ],
4045                         "links": [
4046                             {
4047                                 "href": "/switch",
4048                                 "rt": ["oic.r.switch.binary"],
4049                                 "if": ["oic.if.a", "oic.if.baseline"],
4050                                 "eps": [
4051                                     {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
4052                                     {"ep": "coaps://[fe80::b1d6]:1122"},
4053                                     {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
4054                                 ]
4055                             },
4056                             {
4057                                 "href": "/airFlow",
4058                                 "rt": ["oic.r.airflow"],
4059                                 "if": ["oic.if.a", "oic.if.baseline"],

```

```

4060         "eps": [
4061             { "ep": "coap://[fe80::b1d6]:1111", "pri": 2 },
4062             { "ep": "coaps://[fe80::b1d6]:1122" },
4063             { "ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3 }
4064         ]
4065     }
4066 ]
4067 },
4068 "schema": {
4069     "$ref": "#/definitions/sbaseline"
4070 }
4071 }
4072 },
4073 },
4074 "post": {
4075     "description": "Update on Baseline OCF Interface\n",
4076     "parameters": [
4077         {
4078             "$ref": "#/parameters/interface-update"
4079         },
4080         {
4081             "name": "body",
4082             "in": "body",
4083             "required": true,
4084             "schema": {
4085                 "$ref": "#/definitions/sbaseline-update"
4086             }
4087         }
4088     ],
4089     "responses": {
4090         "200": {
4091             "description": "",
4092             "schema": {
4093                 "$ref": "#/definitions/sbaseline"
4094             }
4095         }
4096     }
4097 },
4098 },
4099 "/CollectionResURI?if=oic.if.b" : {
4100     "get": {
4101         "description": "Collection Resource Type contains Properties and Links.\nThe oic.if.b OCF
4102 Interface exposes a composite representation of the\nResources pointed to by the Links\n",
4103         "parameters": [
4104             {
4105                 "$ref": "#/parameters/interface-all"
4106             }
4107         ],
4108         "responses": {
4109             "200": {
4110                 "description": "All targets returned OK status",
4111                 "x-example": [
4112                     {
4113                         "href": "/switch",
4114                         "rep": {
4115                             "value": true
4116                         }
4117                     },
4118                     {
4119                         "href": "/airFlow",
4120                         "rep": {
4121                             "direction": "floor",
4122                             "speed": 3
4123                         }
4124                     }
4125                 ],
4126                 "schema": {
4127                     "$ref": "#/definitions/sbatch-retrieve"
4128                 }
4129             },
4130             "404": {

```

```

4131         "description" : "One or more targets did not return an OK status, return a
4132 representation containing returned Properties from the targets that returned OK",
4133         "x-example": [
4134             {
4135                 "href": "/switch",
4136                 "rep": {
4137                     "value": true
4138                 }
4139             }
4140         ],
4141         "schema": {
4142             "$ref": "#/definitions/sbatch-retrieve"
4143         }
4144     }
4145 },
4146 "post": {
4147     "description": "Update on Batch OCF Interface\n",
4148     "parameters": [
4149         {
4150             "$ref": "#/parameters/interface-update"
4151         },
4152         {
4153             "name": "body",
4154             "in": "body",
4155             "required": true,
4156             "schema": {
4157                 "$ref": "#/definitions/sbatch-update"
4158             },
4159             "x-example": [
4160                 {
4161                     "href": "/switch",
4162                     "rep": {
4163                         "value": true
4164                     }
4165                 },
4166                 {
4167                     "href": "/airFlow",
4168                     "rep": {
4169                         "direction": "floor",
4170                         "speed": 3
4171                     }
4172                 }
4173             ]
4174         }
4175     ],
4176     "responses": {
4177         "200": {
4178             "description" : "All targets returned OK status, return a representation of the current
4179 state of all targets",
4180             "x-example": [
4181                 {
4182                     "href": "/switch",
4183                     "rep": {
4184                         "value": true
4185                     }
4186                 },
4187                 {
4188                     "href": "/airFlow",
4189                     "rep": {
4190                         "direction": "demist",
4191                         "speed": 5
4192                     }
4193                 }
4194             ],
4195             "schema": {
4196                 "$ref": "#/definitions/sbatch-retrieve"
4197             }
4198         },
4199         "403": {
4200             "description" : "One or more targets did not return OK status; return a retrieve
4201

```

```

4202 representation of the current state of all targets in the batch",
4203     "x-example": [
4204         {
4205             "href": "/switch",
4206             "rep": {
4207                 "value": true
4208             }
4209         },
4210         {
4211             "href": "/airFlow",
4212             "rep": {
4213                 "direction": "floor",
4214                 "speed": 3
4215             }
4216         }
4217     ],
4218     "schema": {
4219         "$ref": "#/definitions/sbatch-retrieve"
4220     }
4221 }
4222 }
4223 }
4224 }
4225 },
4226 "parameters": {
4227     "interface-all" : {
4228         "in" : "query",
4229         "name" : "if",
4230         "type" : "string",
4231         "enum" : ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
4232     },
4233     "interface-update" : {
4234         "in" : "query",
4235         "name" : "if",
4236         "type" : "string",
4237         "enum" : ["oic.if.b", "oic.if.baseline"]
4238     }
4239 },
4240 "definitions": {
4241     "sbaseline" : {
4242         "properties": {
4243             "links" : {
4244                 "description": "A set of simple or individual Links.",
4245                 "items": {
4246                     "$ref": "#/definitions/oic.oic-link"
4247                 },
4248                 "type": "array"
4249             },
4250             "n": {
4251                 "$ref" :
4252 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4253 schema.json#/definitions/n"
4254             },
4255             "id": {
4256                 "$ref" :
4257 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4258 schema.json#/definitions/id"
4259             },
4260             "rt": {
4261                 "$ref": "#/definitions/oic.core.rt-col"
4262             },
4263             "rts": {
4264                 "$ref": "#/definitions/oic.core.rt"
4265             },
4266             "rts-m": {
4267                 "$ref": "#/definitions/oic.core.rt"
4268             },
4269             "if": {
4270                 "description": "The OCF Interfaces supported by this Resource",
4271                 "items": {
4272                     "enum": [

```



```

4273         "oic.if.ll",
4274         "oic.if.baseline",
4275         "oic.if.b"
4276     ],
4277     "type": "string",
4278     "maxLength": 64
4279 },
4280     "minItems": 2,
4281     "uniqueItems": true,
4282     "readOnly": true,
4283     "type": "array"
4284 }
4285 },
4286 "additionalProperties": true,
4287 "type": "object",
4288 "required": [
4289     "rt",
4290     "if",
4291     "links"
4292 ],
4293 },
4294 "sbaseline-update": {
4295     "additionalProperties": true
4296 },
4297     "oic.core.rt-col": {
4298         "description": "Resource Type of the Resource",
4299         "items": {
4300             "enum": ["oic.wk.col"],
4301             "type": "string",
4302             "maxLength": 64
4303         },
4304         "minItems": 1,
4305         "uniqueItems": true,
4306         "readOnly": true,
4307         "type": "array"
4308     },
4309     "oic.core.rt": {
4310         "description": "Resource Type or set of Resource Types",
4311         "items": {
4312             "type": "string",
4313             "maxLength": 64
4314         },
4315         "minItems": 1,
4316         "uniqueItems": true,
4317         "readOnly": true,
4318         "type": "array"
4319     },
4320     "sbatch-retrieve" : {
4321         "minItems" : 1,
4322         "items" : {
4323             "additionalProperties": true,
4324             "properties": {
4325                 "href": {
4326                     "$ref":
4327 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4328 schema.json#/definitions/href"
4329                 },
4330                 "rep": {
4331                     "oneOf": [
4332                         {
4333                             "description": "The response payload from a single Resource",
4334                             "type": "object"
4335                         },
4336                         {
4337                             "description": " The response payload from a Collection (batch) Resource",
4338                             "items": {
4339                                 "$ref": "#/definitions/oic.oic-link"
4340                             },
4341                             "type": "array"
4342                         }
4343                     ]

```

```

4344     }
4345   },
4346   "required": [
4347     "href",
4348     "rep"
4349   ],
4350   "type": "object"
4351 },
4352 "type" : "array"
4353 },
4354 "sbatch-update" : {
4355   "title" : "Collection Batch Update Format",
4356   "minItems" : 1,
4357   "items" : {
4358     "$ref": "#/definitions/sbatch-update.item"
4359   },
4360   "type" : "array"
4361 },
4362 "sbatch-update.item" : {
4363   "additionalProperties": true,
4364   "description": "Array of Resource representations to apply to the batch Collection, using href
4365 to indicate which Resource(s) in the batch to update. If the href Property is empty, effectively
4366 making the URI reference to the Collection itself, the representation is to be applied to all
4367 Resources in the batch",
4368   "properties": {
4369     "href": {
4370       "$ref":
4371 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4372 schema.json#/definitions/href"
4373     },
4374     "rep": {
4375       "oneOf": [
4376         {
4377           "description": "The payload for a single Resource",
4378           "type": "object"
4379         },
4380         {
4381           "description": " The payload for a Collection (batch) Resource",
4382           "items": {
4383             "$ref": "#/definitions/oic.oic-link"
4384           },
4385           "type": "array"
4386         }
4387       ]
4388     }
4389   },
4390   "required": [
4391     "href",
4392     "rep"
4393   ],
4394   "type": "object"
4395 },
4396 "slinks" : {
4397   "type" : "array",
4398   "items" : {
4399     "$ref": "#/definitions/oic.oic-link"
4400   }
4401 },
4402 "oic.oic-link": {
4403   "properties": {
4404     "if": {
4405       "description": "The OCF Interfaces supported by the Linked target",
4406       "items": {
4407         "enum": [
4408           "oic.if.baseline",
4409           "oic.if.ll",
4410           "oic.if.b",
4411           "oic.if.rw",
4412           "oic.if.r",
4413           "oic.if.a",
4414           "oic.if.s"

```

```

4415         ],
4416         "type": "string",
4417         "maxLength": 64
4418     },
4419     "minItems": 1,
4420     "uniqueItems": true,
4421     "readOnly": true,
4422     "type": "array"
4423 },
4424 "rt": {
4425     "$ref": "#/definitions/oic.core.rt"
4426 },
4427 "anchor": {
4428     "$ref":
4429 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4430 schema.json#/definitions/anchor"
4431 },
4432 "di": {
4433     "$ref":
4434 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4435 schema.json#/definitions/di"
4436 },
4437 "eps": {
4438     "$ref":
4439 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4440 schema.json#/definitions/eps"
4441 },
4442 "href": {
4443     "$ref":
4444 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4445 schema.json#/definitions/href"
4446 },
4447 "ins": {
4448     "$ref":
4449 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4450 schema.json#/definitions/ins"
4451 },
4452 "p": {
4453     "$ref":
4454 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4455 schema.json#/definitions/p"
4456 },
4457 "rel": {
4458     "$ref":
4459 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4460 schema.json#/definitions/rel_array"
4461 },
4462 "title": {
4463     "$ref":
4464 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4465 schema.json#/definitions/title"
4466 },
4467 "type": {
4468     "$ref":
4469 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4470 schema.json#/definitions/type"
4471 },
4472 "tag-pos-desc": {
4473     "$ref":
4474 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4475 schema.json#/definitions/tag-pos-desc"
4476 },
4477 "tag-pos-rel": {
4478     "$ref":
4479 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4480 schema.json#/definitions/tag-pos-rel"
4481 },
4482 "tag-func-desc": {
4483     "$ref":
4484 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
4485 schema.json#/definitions/tag-func-desc"

```

```

4486     }
4487   },
4488   "required": [
4489     "href",
4490     "rt",
4491     "if"
4492   ],
4493   "type": "object"
4494 }
4495 }
4496 }
4497

```

### A.3.5 Property definition

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

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

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

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

### A.3.6 CRUDN behaviour

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

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

Create	Read	Update	Delete	Notify
	get	post		observe

## A.4 Device

### A.4.1 Introduction

Known Resource that is hosted by every Server.

Allows for logical Device specific information to be discovered.

### A.4.2 Well-known URI

/oic/d

### A.4.3 Resource type

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

### A.4.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Device",
    "version": "2019-03-13",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
}
```

```

4532     "produces": [
4533         "application/json"
4534     ],
4535     "paths": {
4536         "/oic/d" : {
4537             "get": {
4538                 "description": "Known Resource that is hosted by every Server.\nAllows for logical Device
4539 specific information to be discovered.\n",
4540                 "parameters": [
4541                     {
4542                         "$ref": "#/parameters/interface"
4543                     }
4544                 ],
4545                 "responses": {
4546                     "200": {
4547                         "description": "",
4548                         "x-example":
4549                             {
4550                                 "n": "Device 1",
4551                                 "rt": ["oic.wk.d"],
4552                                 "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
4553                                 "icv": "ocf.2.0.2",
4554                                 "dmv": "ocf.res.1.0.0, ocf.sh.1.0.0",
4555                                 "piid": "6F0AAC04-2BB0-468D-B57C-16570A26AE48"
4556                             },
4557                         "schema": {
4558                             "$ref": "#/definitions/Device"
4559                         }
4560                     }
4561                 }
4562             }
4563         }
4564     },
4565     "parameters": {
4566         "interface" : {
4567             "in": "query",
4568             "name": "if",
4569             "type": "string",
4570             "enum": ["oic.if.r", "oic.if.baseline"]
4571         }
4572     },
4573     "definitions": {
4574         "Device": {
4575             "properties": {
4576                 "rt": {
4577                     "description": "Resource Type of the Resource",
4578                     "items": {
4579                         "type": "string",
4580                         "maxLength": 64
4581                     },
4582                     "minItems": 1,
4583                     "readOnly": true,
4584                     "uniqueItems": true,
4585                     "type": "array"
4586                 },
4587                 "ld": {
4588                     "description": "Localized Descriptions.",
4589                     "items": {
4590                         "properties": {
4591                             "language": {
4592                                 "allOf": [
4593                                     {
4594                                         "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4595 schema.json#/definitions/language-tag"
4596                                     },
4597                                     {
4598                                         "description": "An RFC 5646 language tag.",
4599                                         "readOnly": true
4600                                     }
4601                                 ]
4602                             }

```

```

4603         "value": {
4604             "description": "Device description in the indicated language.",
4605             "maxLength": 64,
4606             "readOnly": true,
4607             "type": "string"
4608         }
4609     },
4610     "type": "object"
4611 },
4612 "minItems": 1,
4613 "readOnly": true,
4614 "type": "array"
4615 },
4616 "piid": {
4617     "allOf": [
4618         {
4619             "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4620 schema.json#/definitions/uuid"
4621         },
4622         {
4623             "description": "Protocol independent unique identifier for the Device that is
4624 immutable.",
4625             "readOnly": true
4626         }
4627     ]
4628 },
4629 "di": {
4630     "allOf": [
4631         {
4632             "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4633 schema.json#/definitions/uuid"
4634         },
4635         {
4636             "description": "Unique identifier for the Device",
4637             "readOnly": true
4638         }
4639     ]
4640 },
4641 "dmno": {
4642     "description": "Model number as designated by manufacturer.",
4643     "maxLength": 64,
4644     "readOnly": true,
4645     "type": "string"
4646 },
4647 "sv": {
4648     "description": "Software version.",
4649     "maxLength": 64,
4650     "readOnly": true,
4651     "type": "string"
4652 },
4653 "dmn": {
4654     "description": "Manufacturer Name.",
4655     "items": {
4656         "properties": {
4657             "language": {
4658                 "allOf": [
4659                     {
4660                         "$ref" : "http://openconnectivityfoundation.github.io/core/schemas/oic.types-
4661 schema.json#/definitions/language-tag"
4662                     },
4663                     {
4664                         "description": "An RFC 5646 language tag.",
4665                         "readOnly": true
4666                     }
4667                 ]
4668             },
4669             "value": {
4670                 "description": "Manufacturer name in the indicated language.",
4671                 "maxLength": 64,
4672                 "readOnly": true,
4673                 "type": "string"

```

```

4674         }
4675     },
4676     "type": "object"
4677 },
4678 "minItems": 1,
4679 "readOnly": true,
4680 "type": "array"
4681 },
4682 "icv": {
4683     "description": "The version of the Device",
4684     "maxLength": 64,
4685     "readOnly": true,
4686     "type": "string"
4687 },
4688 "dmv": {
4689     "description": "Specification versions of the Resource and Device Specifications to which
4690 this device data model is implemented",
4691     "maxLength": 256,
4692     "readOnly": true,
4693     "type": "string"
4694 },
4695 "n": {
4696     "$ref": :
4697 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4698 schema.json#/definitions/n"
4699 },
4700 "id": {
4701     "$ref": :
4702 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4703 schema.json#/definitions/id"
4704 },
4705 "if": {
4706     "description": "The OCF Interfaces supported by this Resource",
4707     "items": {
4708         "enum": [
4709             "oic.if.r",
4710             "oic.if.baseline"
4711         ],
4712         "type": "string",
4713         "maxLength": 64
4714     },
4715     "minItems": 2,
4716     "uniqueItems": true,
4717     "readOnly": true,
4718     "type": "array"
4719 },
4720 "econame": {
4721     "description": "Ecosystem Name of the Bridged Device which is exposed by this VOD.",
4722     "type": "string",
4723     "enum": ["BLE", "oneM2M", "UPlus", "Zigbee", "Z-Wave"],
4724     "readOnly": true
4725 },
4726 "ecoversion": {
4727     "description": "Version of ecosystem that a Bridged Device belongs to. Typical version
4728 string format is like n.n (e.g. 5.0).",
4729     "type": "string",
4730     "maxLength": 64,
4731     "readOnly": true
4732 }
4733 },
4734 "type": "object",
4735 "required": ["n", "di", "icv", "dmv", "piid"]
4736 }
4737 }
4738 }
4739

```

#### A.4.5 Property definition

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



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

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

**A.4.6 CRUDN behaviour**

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

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

Create	Read	Update	Delete	Notify
	get			observe

## A.5 Introspection Resource

### A.5.1 Introduction

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

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

### A.5.2 Well-known URI

/IntrospectionResURI

### A.5.3 Resource type

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

### A.5.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Introspection Resource",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/IntrospectionResURI": {
      "get": {
        "description": "This Resource provides the means to get the Introspection Device Data (IDD) specifying all the OCF Endpoints of the Device.\nThe url hosted by this Resource is either a local or an external url.\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface"
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": {
              "rt": ["oic.wk.introspection"],
              "urlInfo": [
                {
                  "content-type": "application/cbor",
                  "protocol": "coap",
                  "url": "coap://[fe80::1]:1234/IntrospectionExampleURI"
                }
              ]
            }
          }
        ],
        "schema": {
          "$ref": "#/definitions/oic.wk.introspectionInfo"
        }
      }
    }
  }
}
```

```

4807     }
4808   }
4809 },
4810 "parameters": {
4811   "interface": {
4812     "in": "query",
4813     "name": "if",
4814     "type": "string",
4815     "enum": ["oic.if.r", "oic.if.baseline"]
4816   }
4817 },
4818 "definitions": {
4819   "oic.wk.introspectionInfo": {
4820     "properties": {
4821       "rt": {
4822         "description": "Resource Type of the Resource",
4823         "items": {
4824           "enum": ["oic.wk.introspection"],
4825           "type": "string",
4826           "maxLength": 64
4827         },
4828         "minItems": 1,
4829         "readOnly": true,
4830         "uniqueItems": true,
4831         "type": "array"
4832       },
4833       "n": {
4834         "$ref":
4835 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4836 schema.json#/definitions/n"
4837       },
4838       "urlInfo": {
4839         "description": "Information on the location of the Introspection Device Data (IDD).",
4840         "items": {
4841           "properties": {
4842             "content-type": {
4843               "default": "application/cbor",
4844               "description": "content-type of the Introspection Device Data",
4845               "enum": [
4846                 "application/json",
4847                 "application/cbor"
4848               ],
4849               "type": "string"
4850             },
4851             "protocol": {
4852               "description": "Identifier for the protocol to be used to obtain the Introspection
4853 Device Data",
4854               "enum": [
4855                 "coap",
4856                 "coaps",
4857                 "http",
4858                 "https",
4859                 "coap+tcp",
4860                 "coaps+tcp"
4861               ],
4862               "type": "string"
4863             },
4864             "url": {
4865               "description": "The URL of the Introspection Device Data.",
4866               "format": "uri",
4867               "type": "string"
4868             },
4869             "version": {
4870               "default": 1,
4871               "description": "The version of the Introspection Device Data that can be
4872 downloaded",
4873               "enum": [
4874                 1
4875               ],
4876               "type": "integer"
4877             }

```

```

4878         },
4879         "required": [
4880             "url",
4881             "protocol"
4882         ],
4883         "type": "object"
4884     },
4885     "minItems": 1,
4886     "readOnly": true,
4887     "type": "array"
4888 },
4889 "id": {
4890     "$ref":
4891 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
4892 schema.json#/definitions/id"
4893 },
4894 "if": {
4895     "description": "The OCF Interfaces supported by this Resource",
4896     "items": {
4897         "enum": [
4898             "oic.if.r",
4899             "oic.if.baseline"
4900         ],
4901         "type": "string",
4902         "maxLength": 64
4903     },
4904     "minItems": 2,
4905     "readOnly": true,
4906     "uniqueItems": true,
4907     "type": "array"
4908 }
4909 },
4910 "type" : "object",
4911 "required": ["urlInfo"]
4912 }
4913 }
4914 }
4915

```

#### 4916 A.5.5 Property definition

4917 Table A.8 defines the Properties that are part of the "oic.wk.introspection" Resource Type.

4918 **Table A.8 – The Property definitions of the Resource with type "rt" =**  
4919 **"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

#### 4920 A.5.6 CRUDN behaviour

4921 Table A.9 defines the CRUDN operations that are supported on the "oic.wk.introspection" Resource  
4922 Type.

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

Create	Read	Update	Delete	Notify
	get			observe

## A.6 Platform

### A.6.1 Introduction

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

### A.6.2 Well-known URI

/oic/p

### A.6.3 Resource type

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

### A.6.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Platform",
    "version": "2019-03-04",
    "license": {
      "name": "OCF Data Model License",
      "url":
"https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
CENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": ["http"],
  "consumes": ["application/json"],
  "produces": ["application/json"],
  "paths": {
    "/oic/p" : {
      "get": {
        "description": "Known Resource that is defines the Platform on which an Server is
hosted.\nAllows for Platform specific information to be discovered.\n",
        "parameters": [
          {"$ref": "#/parameters/interface"}
        ],
        "responses": {
          "200": {
            "description" : "",
            "x-example": {
              "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
              "rt": ["oic.wk.p"],
              "mnmn": "Acme, Inc"
            },
            "schema": { "$ref": "#/definitions/Platform" }
          }
        }
      }
    }
  },
  "parameters": {
    "interface" : {
      "in" : "query",
      "name" : "if",
      "type" : "string",
      "enum" : ["oic.if.r", "oic.if.baseline"]
    }
  }
}
```

```

4979     }
4980   },
4981   "definitions": {
4982     "Platform": {
4983       "properties": {
4984         "rt": {
4985           "description": "Resource Type of the Resource",
4986           "items": {
4987             "enum": ["oic.wk.p"],
4988             "type": "string",
4989             "maxLength": 64
4990           },
4991           "minItems": 1,
4992           "uniqueItems": true,
4993           "readOnly": true,
4994           "type": "array"
4995         },
4996         "pi": {
4997           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
4998 9]{12}$",
4999           "type": "string",
5000           "description": "Platform Identifier",
5001           "readOnly": true
5002         },
5003         "mnfv": {
5004           "description": "Manufacturer's firmware version",
5005           "maxLength": 64,
5006           "readOnly": true,
5007           "type": "string"
5008         },
5009         "vid": {
5010           "description": "Manufacturer's defined information for the Platform. The content is
5011 freeform, with population rules up to the manufacturer",
5012           "maxLength": 64,
5013           "readOnly": true,
5014           "type": "string"
5015         },
5016         "mnmn": {
5017           "description": "Manufacturer name",
5018           "maxLength": 64,
5019           "readOnly": true,
5020           "type": "string"
5021         },
5022         "mnmo": {
5023           "description": "Model number as designated by the manufacturer",
5024           "maxLength": 64,
5025           "readOnly": true,
5026           "type": "string"
5027         },
5028         "mnhw": {
5029           "description": "Platform Hardware Version",
5030           "maxLength": 64,
5031           "readOnly": true,
5032           "type": "string"
5033         },
5034         "mnos": {
5035           "description": "Platform Resident OS Version",
5036           "maxLength": 64,
5037           "readOnly": true,
5038           "type": "string"
5039         },
5040         "mndt": {
5041           "pattern": "^[([0-9]{4})-(1[0-2]|0[1-9])-(3[0-1]|2[0-9]|1[0-9]|0[1-9])$",
5042           "type": "string",
5043           "description": "Manufacturing Date.",
5044           "readOnly": true
5045         },
5046         "id": {
5047           "$ref":
5048 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5049 schema.json#/definitions/id"

```

```

5050     },
5051     "mnsi" : {
5052         "description": "Manufacturer's Support Information URL",
5053         "format": "uri",
5054         "maxLength": 256,
5055         "readOnly": true,
5056         "type": "string"
5057     },
5058     "mnpv" : {
5059         "description": "Platform Version",
5060         "maxLength": 64,
5061         "readOnly": true,
5062         "type": "string"
5063     },
5064     "st" : {
5065         "description": "The date-time format pattern according to IETF RFC 3339.",
5066         "format": "date-time",
5067         "readOnly": true,
5068         "type": "string"
5069     },
5070     "n" : {
5071         "$ref":
5072         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5073         schema.json#/definitions/n"
5074     },
5075     "mnml" : {
5076         "description": "Manufacturer's URL",
5077         "format": "uri",
5078         "maxLength": 256,
5079         "readOnly": true,
5080         "type": "string"
5081     },
5082     "mnsel" : {
5083         "description": "Serial number as designated by the manufacturer",
5084         "maxLength": 64,
5085         "readOnly": true,
5086         "type": "string"
5087     },
5088     "if" : {
5089         "description": "The OCF Interfaces supported by this Resource",
5090         "items": {
5091             "enum": [
5092                 "oic.if.r",
5093                 "oic.if.baseline"
5094             ],
5095             "type": "string",
5096             "maxLength": 64
5097         },
5098         "minItems": 2,
5099         "readOnly": true,
5100         "uniqueItems": true,
5101         "type": "array"
5102     },
5103     "mnct" : {
5104         "description": "An array of integers and each integer indicates the network connectivity
5105         type based on IANAIfType value as defined by: https://www.iana.org/assignments/ianaiftype-
5106         mib/ianaiftype-mib, e.g., [71, 259] which represents Wi-Fi and Zigbee.",
5107         "items": {
5108             "type": "integer",
5109             "minimum": 1,
5110             "description": "The network connectivity type based on IANAIfType value as defined by:
5111             https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib."
5112         },
5113         "minItems": 1,
5114         "readOnly": true,
5115         "type": "array"
5116     }
5117 },
5118 "type" : "object",
5119 "required": ["pi", "mnmn"]
5120 }

```

5121 }  
 5122 }  
 5123 }

## 5124 A.6.5 Property definition

5125 Table A.10 defines the Properties that are part of the "oic.wk.p" Resource Type.

5126 **Table A.10 – The Property definitions of the Resource with type "rt" = "oic.wk.p".**

Property name	Value type	Mandatory	Access mode	Description
rt	array: see schema	No	Read Only	Resource Type of the Resource
pi	string	Yes	Read Only	Platform Identifier
mnfv	string	No	Read Only	Manufacturer's firmware version
vid	string	No	Read Only	Manufacturer's defined information for the Platform. The content is freeform, with population rules up to the manufacturer
mnmn	string	Yes	Read Only	Manufacturer name
mnmo	string	No	Read Only	Model number as designated by the manufacturer
mnhw	string	No	Read Only	Platform Hardware Version
mnos	string	No	Read Only	Platform Resident OS Version
mndt	string	No	Read Only	Manufacturing Date.
id	multiple types: see schema	No	Read Write	
mnsi	string	No	Read Only	Manufacturer's Support Information URL
mnpv	string	No	Read Only	Platform Version
st	string	No	Read Only	The date-time format pattern according to IETF RFC 3339.
n	multiple types: see schema	No	Read Write	
mnml	string	No	Read Only	Manufacturer's URL
mnsel	string	No	Read Only	Serial number as designated by the manufacturer
if	array: see schema	No	Read Only	The OCF Interfaces supported by this Resource
mnct	array: see schema	No	Read Only	An array of integers and each integer indicates the network connectivity type based on IANAIfType value as defined by: <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.

## 5127 A.6.6 CRUDN behaviour

5128 Table A.11 defines the CRUDN operations that are supported on the "oic.wk.p" Resource Type.

5129 **Table A.11 – The CRUDN operations of the Resource with type "rt" = "oic.wk.p".**

Create	Read	Update	Delete	Notify
	get			observe



## A.7 Discoverable Resources

### A.7.1 Introduction

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

### A.7.2 Well-known URI

/oic/res

### A.7.3 Resource type

The Resource Type is defined as: "oic.wk.res".

### A.7.4 OpenAPI 2.0 definition

```
{
  "swagger": "2.0",
  "info": {
    "title": "Discoverable Resources",
    "version": "2019-04-22",
    "license": {
      "name": "OCF Data Model License",
      "url": "https://openconnectivityfoundation.github.io/core/LICENSE.md",
      "x-copyright": "Copyright 2016-2019 Open Connectivity Foundation, Inc. All rights reserved."
    },
    "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
  },
  "schemes": [
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/oic/res?if=oic.if.ll": {
      "get": {
        "description": "Links list representation of /oic/res; list of discoverable Resources\n",
        "parameters": [
          {
            "$ref": "#/parameters/interface-all"
          }
        ],
        "responses": {
          "200": {
            "description": "",
            "x-example": [
              {
                "href": "/oic/res",
                "rt": ["oic.wk.res"],
                "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
                "rel": ["self"],
                "p": {"bm": 3},
                "eps": [
                  {"ep": "coaps://[fe80::bld6]:1122"}
                ]
              },
              {
                "href": "/humidity",
                "rt": ["oic.r.humidity"],
                "if": ["oic.if.s", "oic.if.baseline"],
                "p": {"bm": 3},
                "eps": [
                  {"ep": "coaps://[fe80::bld6]:1111", "pri": 2},
                  {"ep": "coaps://[fe80::bld6]:1122"},
                  {"ep": "coaps+tcp://[2001:db8:a::123]:2222", "pri": 3}
                ]
              }
            ]
          }
        }
      }
    }
  }
}
```

```

5192         },
5193         {
5194             "href": "/temperature",
5195             "rt": ["oic.r.temperature"],
5196             "if": ["oic.if.s", "oic.if.baseline"],
5197             "p": {"bm": 3},
5198             "eps": [
5199                 {"ep": "coaps://[[2001:db8:a::123]:2222"}
5200             ]
5201         }
5202     ],
5203     "schema": {
5204         "$ref": "#/definitions/slinklist"
5205     }
5206 }
5207 }
5208 },
5209 },
5210 "/oic/res?if=oic.if.b" : {
5211     "get": {
5212         "description": "Batch representation of /oic/res; list of discoverable Resources\n",
5213         "parameters": [
5214             {"$ref": "#/parameters/interface-all"}
5215         ],
5216         "responses": {
5217             "200": {
5218                 "description": "",
5219                 "x-example": [
5220                     {
5221                         "href": "/humidity",
5222                         "rep": {
5223                             "rt": ["oic.r.humidity"],
5224                             "humidity": 40,
5225                             "desiredHumidity": 40
5226                         }
5227                     },
5228                     {
5229                         "href": "/temperature",
5230                         "rep": {
5231                             "rt": ["oic.r.temperature"],
5232                             "temperature": 20.0,
5233                             "units": "C"
5234                         }
5235                     }
5236                 ],
5237                 "schema": {"$ref": "#/definitions/sbatch"}
5238             }
5239         }
5240     },
5241 },
5242 "/oic/res?if=oic.if.baseline": {
5243     "get": {
5244         "description": "Baseline representation of /oic/res; list of discoverable Resources\n",
5245         "parameters": [
5246             {"$ref": "#/parameters/interface-all"}
5247         ],
5248     },
5249     "responses": {
5250         "200": {
5251             "description": "",
5252             "x-example": [
5253                 {
5254                     "rt": ["oic.wk.res"],
5255                     "if": ["oic.if.ll", "oic.if.b", "oic.if.baseline"],
5256                     "links": [
5257                         {
5258                             "href": "/humidity",
5259                             "rt": ["oic.r.humidity"],
5260                             "if": ["oic.if.s", "oic.if.baseline"],
5261                             "p": {"bm": 3},

```

```

5263         "eps": [
5264             {"ep": "coaps://[fe80::b1d6]:1111", "pri": 2},
5265             {"ep": "coaps://[fe80::b1d6]:1122"},
5266             {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5267         ]
5268     },
5269     {
5270         "href": "/temperature",
5271         "rt": ["oic.r.temperature"],
5272         "if": ["oic.if.s", "oic.if.baseline"],
5273         "p": {"bm": 3},
5274         "eps": [
5275             {"ep": "coaps://[[2001:db8:a::123]:2222"}
5276         ]
5277     }
5278 ]
5279 },
5280 ],
5281 "schema": {
5282     "$ref": "#/definitions/sbaseline"
5283 },
5284 },
5285 },
5286 },
5287 },
5288 },
5289 "parameters": {
5290     "interface-all": {
5291         "in": "query",
5292         "name": "if",
5293         "type": "string",
5294         "enum": ["oic.if.ll", "oic.if.b", "oic.if.baseline"]
5295     }
5296 },
5297 "definitions": {
5298     "oic.oic-link": {
5299         "type": "object",
5300         "properties": {
5301             "anchor": {
5302                 "$ref":
5303 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5304 schema.json#/definitions/anchor"
5305             },
5306             "di": {
5307                 "$ref":
5308 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5309 schema.json#/definitions/di"
5310             },
5311             "eps": {
5312                 "$ref":
5313 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5314 schema.json#/definitions/eps"
5315             },
5316             "href": {
5317                 "$ref":
5318 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5319 schema.json#/definitions/href"
5320             },
5321             "if": {
5322                 "description": "The OCF Interfaces supported by the Linked Resource",
5323                 "items": {
5324                     "enum": [
5325                         "oic.if.baseline",
5326                         "oic.if.ll",
5327                         "oic.if.b",
5328                         "oic.if.rw",
5329                         "oic.if.r",
5330                         "oic.if.a",
5331                         "oic.if.s"
5332                     ],
5333                     "type": "string",

```

```

5334         "maxLength": 64
5335     },
5336     "minItems": 1,
5337     "uniqueItems": true,
5338     "type": "array"
5339 },
5340 "ins": {
5341     "$ref":
5342 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5343 schema.json#/definitions/ins"
5344 },
5345 "p": {
5346     "$ref":
5347 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5348 schema.json#/definitions/p"
5349 },
5350 "rel": {
5351     "description": "The relation of the target URI referenced by the Link to the context URI",
5352     "oneOf": [
5353         {
5354             "$ref":
5355 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5356 schema.json#/definitions/rel_array"
5357         },
5358         {
5359             "$ref":
5360 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5361 schema.json#/definitions/rel_string"
5362         }
5363     ]
5364 },
5365 "rt": {
5366     "description": "Resource Type of the Linked Resource",
5367     "items": {
5368         "maxLength": 64,
5369         "type": "string"
5370     },
5371     "minItems": 1,
5372     "uniqueItems": true,
5373     "type": "array"
5374 },
5375 "title": {
5376     "$ref":
5377 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5378 schema.json#/definitions/title"
5379 },
5380 "type": {
5381     "$ref":
5382 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5383 schema.json#/definitions/type"
5384 },
5385 "tag-pos-desc": {
5386     "$ref":
5387 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5388 schema.json#/definitions/tag-pos-desc"
5389 },
5390 "tag-pos-rel": {
5391     "$ref":
5392 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5393 schema.json#/definitions/tag-pos-rel"
5394 },
5395 "tag-func-desc": {
5396     "$ref":
5397 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5398 schema.json#/definitions/tag-func-desc"
5399 }
5400 },
5401 "required": [
5402     "href",
5403     "rt",
5404     "if"

```

```

5405     ]
5406   },
5407   "slinklist": {
5408     "type": "array",
5409     "readOnly": true,
5410     "items": {
5411       "$ref": "#/definitions/oic.oic-link"
5412     }
5413   },
5414   "sbaseline": {
5415     "type": "array",
5416     "minItems": 1,
5417     "maxItems": 1,
5418     "items": {
5419       "type": "object",
5420       "properties": {
5421         "n": {
5422           "$ref":
5423 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5424 schema.json#/definitions/n"
5425         },
5426         "id": {
5427           "$ref":
5428 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5429 schema.json#/definitions/id"
5430         },
5431         "rt": {
5432           "description": "Resource Type of this Resource",
5433           "items": {
5434             "enum": ["oic.wk.res"],
5435             "type": "string",
5436             "maxLength": 64
5437           },
5438           "minItems": 1,
5439           "readOnly": true,
5440           "uniqueItems": true,
5441           "type": "array"
5442         },
5443         "if": {
5444           "description": "The OCF Interfaces supported by this Resource",
5445           "items": {
5446             "enum": [
5447               "oic.if.ll",
5448               "oic.if.b",
5449               "oic.if.baseline"
5450             ],
5451             "type": "string",
5452             "maxLength": 64
5453           },
5454           "minItems": 2,
5455           "readOnly": true,
5456           "uniqueItems": true,
5457           "type": "array"
5458         },
5459         "links": {
5460           "type": "array",
5461           "items": {
5462             "$ref": "#/definitions/oic.oic-link"
5463           }
5464         },
5465         "sduuid": {
5466           "description": "A UUID that identifies the Security Domain.",
5467           "type": "string",
5468           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5469 9]{12}$",
5470           "readOnly": true
5471         },
5472         "sdname": {
5473           "description": "Human-friendly name for the Security Domain.",
5474           "type": "string",
5475           "readOnly": true

```

```

5476     }
5477   },
5478   "required": [
5479     "rt",
5480     "if",
5481     "links"
5482   ]
5483 },
5484 {
5485   "sbatch" : {
5486     "type" : "array",
5487     "minItems" : 1,
5488     "items" : {
5489       "type": "object",
5490       "additionalProperties": true,
5491       "properties": {
5492         "href": {
5493           "$ref":
5494 "https://openconnectivityfoundation.github.io/core/schemas/oic.links.properties.core-
5495 schema.json#/definitions/href"
5496         },
5497         "rep": {
5498           "oneOf": [
5499             {
5500               "description": "The response payload from a single Resource",
5501               "type": "object"
5502             },
5503             {
5504               "description": " The response payload from a Collection (batch) Resource",
5505               "items": {
5506                 "$ref": "#/definitions/oic.oic-link"
5507               },
5508               "type": "array"
5509             }
5510           ]
5511         }
5512       },
5513       "required": [
5514         "href",
5515         "rep"
5516       ]
5517     }
5518   }
5519 }
5520 }
5521

```

## 5522 A.7.5 Property definition

5523 Table A.12 defines the Properties that are part of the "oic.wk.res" Resource Type.

5524 **Table A.12 – The Property definitions of the Resource with type "rt" = "oic.wk.res".**

Property name	Value type	Mandatory	Access mode	Description
anchor	multiple types: see schema	No	Read Write	
di	multiple types: see schema	No	Read Write	
eps	multiple types: see schema	No	Read Write	
href	multiple types: see schema	Yes	Read Write	
if	array: see schema	Yes	Read Write	The OCF Interfaces supported by the Linked Resource

ins	multiple types: see schema	No	Read Write	
p	multiple types: see schema	No	Read Write	
rel	multiple types: see schema	No	Read Write	The relation of the target URI referenced by the Link to the context URI
rt	array: see schema	Yes	Read Write	Resource Type of the Linked Resource
title	multiple types: see schema	No	Read Write	
type	multiple types: see schema	No	Read Write	
tag-pos-desc	multiple types: see schema	No	Read Write	
tag-pos-rel	multiple types: see schema	No	Read Write	
tag-func-desc	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	Yes	Read Only	Resource Type of this Resource
if	array: see schema	Yes	Read Only	The OCF Interfaces supported by this Resource
links	array: see schema	Yes	Read Write	
sduuid	string	No	Read Only	A UUID that identifies the Security Domain.
sdname	string	No	Read Only	Human-friendly name for the Security Domain.
href	multiple types: see schema	Yes	Read Write	
rep	multiple types: see schema	Yes	Read Write	

#### A.7.6 CRUDN behaviour

Table A.13 defines the CRUDN operations that are supported on the "oic.wk.res" Resource Type.

**Table A.13 – The CRUDN operations of the Resource with type "rt" = "oic.wk.res".**

Create	Read	Update	Delete	Notify
	get			observe

Annex B  
(informative)

**OpenAPI 2.0 Schema Extension**

**B.1 OpenAPI 2.0 Schema Reference**

OpenAPI 2.0 does not support allOf and anyOf JSON schema valiation constructs; this document has extended the underlying OpenAPI 2.0 schema to enable these, all OpenAPI 2.0 files are valid against the extended schema. Reference the following location for a copy of the extended schema:

- <https://github.com/openconnectivityfoundation/OCFswagger2.0-schema>

**B.2 OpenAPI 2.0 Introspection empty file**

Reference the following location for a copy of an empty OpenAPI 2.0 file:

- <https://github.com/openconnectivityfoundation/DeviceBuilder/blob/master/introspection-examples/introspection-empty.txt>



5543  
5544  
5545  
5546

Annex C  
(normative)

Semantic Tag enumeration support

5547

C.1 Introduction

5548

This Annex defines the enumerations that are applicable to defined Semantic Tags.

5549

C.2 "tag-pos-desc" supported enumeration

5550

5551

Figure C.1 defines the enumeration from which a value populated within an instance of the "tag-pos-desc" Semantic Tag is taken.

```
"pos-descriptions": {
  "enum":
  [ "unknown", "top", "bottom", "left", "right", "centre", "topleft", "bottomleft", "centreleft",
    "centreright", "bottomright", "topright", "topcentre", "bottomcentre" ]
}
```

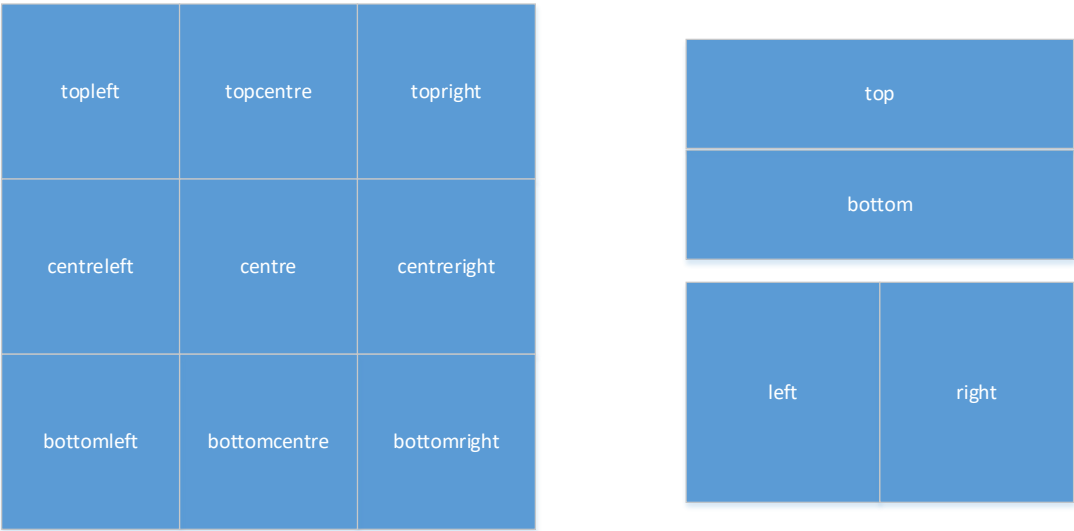
5552

Figure C.1 – Enumeration for "tag-pos-desc" Semantic Tag

5553

5554

Figure C.2 provides an illustrative representation of the definition of the values that can be represented within an instance of "tag-pos-desc".



5555

5556

Figure C.2 – Definition of "tag-pos-desc" Semantic Tag values

5557

C.3 "tag-loc" supported enumeration

5558

5559

Figure C.3 defines the enumeration from which a value populated within an instance of the "tag-locn" Semantic Tag is taken.

```
"locn-descriptions": {
  "enum":
  [ "unknown", "attic", "balcony", "ballroom", "bathroom", "bedroom", "border", "boxroom", "cellar", "cloakr
    oom", "conservatory", "corridor", "deck", "den", "diningroom", "drawingroom", "driveway", "dungeon", "ens
```

```
uite","entrance","familyroom","garage","garden","guestroom","hall","indoor","kitchen","larder","  
lawn","library","livingroom","lounge","mancafe","masterbedroom","musicroom","office","outdoor","  
pantry","parkinglot","parlour","patio","receptionroom","restroom","roof","roofterrace","sauna",  
"scullery","shed","sittingroom","snug","spa","studio","suite","swimmingpool","terrace","toilet",  
"utilityroom","vegetableplot","ward","yard"]  
  
}
```

**Figure C.3 – Enumeration for "tag-locn" Semantic Tag**

5562

## Bibliography

- 5563 [1] OCF Core - Optional, Information technology – Open Connectivity Foundation (OCF)  
5564 Specification – Part X: Core - Optional specification  
5565 Latest version available at:  
5566 [https://openconnectivity.org/specs/OCF\\_Core\\_Optional\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Core_Optional_Specification.pdf)
- 5567 [2] OCF Easy Wi-Fi Setup, Information technology – Open Connectivity Foundation (OCF)  
5568 Specification – Part 7: Wi-Fi Easy Setup specification  
5569 Latest version available at: [https://openconnectivity.org/specs/OCF\\_Wi-](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)  
5570 [Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)  
5571