



# OIC CORE SPECIFICATION V1.1.2 Part 1

Open Connectivity Foundation (OCF)  
[admin@openconnectivity.org](mailto:admin@openconnectivity.org)

## Legal Disclaimer

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

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

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

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

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

# CONTENTS

22

23

|    |       |   |    |
|----|-------|---|----|
| 24 | 1     | Scope .....   | 11 |
| 25 | 2     | Normative references .....                            | 11 |
| 26 | 3     | Terms, definitions, symbols and abbreviations .....   | 14 |
| 27 | 3.1   | Terms and definitions .....                           | 14 |
| 28 | 3.2   | Symbols and abbreviations .....                       | 16 |
| 29 | 3.3   | Conventions .....                                     | 17 |
| 30 | 3.4   | Data types .....                                      | 17 |
| 31 | 4     | Document conventions and organization .....           | 18 |
| 32 | 5     | Architecture.....                                     | 19 |
| 33 | 5.1   | Overview .....  | 19 |
| 34 | 5.2   | Principle .....                                       | 19 |
| 35 | 5.3   | Functional block diagram.....                         | 21 |
| 36 | 5.3.1 | Framework .....                                       | 22 |
| 37 | 5.4   | Example Scenario with roles.....                      | 22 |
| 38 | 5.5   | Example Scenario: Bridging to Non- OCF ecosystem..... | 23 |
| 39 | 6     | Identification and addressing.....                    | 24 |
| 40 | 6.1   | Introduction .....                                    | 24 |
| 41 | 6.2   | Identification.....                                   | 25 |
| 42 | 6.2.1 | Resource identification and addressing .....          | 25 |
| 43 | 6.3   | Namespace: .....                                      | 26 |
| 44 | 6.4   | Network addressing .....                              | 26 |
| 45 | 7     | Resource model .....                                  | 26 |
| 46 | 7.1   | Introduction .....                                    | 26 |
| 47 | 7.2   | Resource.....   | 27 |
| 48 | 7.3   | Property .....  | 28 |
| 49 | 7.3.1 | Introduction .....                                    | 28 |
| 50 | 7.3.2 | Common Properties.....                                | 29 |
| 51 | 7.4   | Resource Type .....                                   | 30 |
| 52 | 7.4.1 | Introduction .....                                    | 30 |
| 53 | 7.4.2 | Resource Type Property.....                           | 31 |
| 54 | 7.4.3 | Resource Type definition.....                         | 31 |
| 55 | 7.5   | Device Type .....                                     | 32 |
| 56 | 7.6   | Interface .....                                       | 33 |
| 57 | 7.6.1 | Introduction .....                                    | 33 |
| 58 | 7.6.2 | Interface Property .....                              | 33 |
| 59 | 7.6.3 | Interface methods .....                               | 34 |
| 60 | 7.7   | Resource representation .....                         | 42 |
| 61 | 7.8   | Structure .....                                       | 42 |
| 62 | 7.8.1 | Introduction .....                                    | 42 |
| 63 | 7.8.2 | Resource Relationships.....                           | 42 |
| 64 | 7.8.3 | Collections .....                                     | 48 |

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

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

|     |       |  |     |
|-----|-------|--|-----|
| 152 | D.2.3 | Resource Type .....                              | 108 |
| 153 | D.2.4 | RAML Definition .....                            | 109 |
| 154 | D.2.5 | Property Definition .....                        | 113 |
| 155 | D.2.6 | CRUDN behavior .....                             | 114 |
| 156 | D.2.7 | Referenced JSON schemas .....                    | 115 |
| 157 | D.2.8 | oic.oic-link-schema.json .....                   | 115 |
| 158 | D.3   | OIC Configuration .....                          | 117 |
| 159 | D.3.1 | Introduction .....                               | 117 |
| 160 | D.3.2 | Example URI .....                                | 117 |
| 161 | D.3.3 | Resource Type .....                              | 117 |
| 162 | D.3.4 | RAML Definition .....                            | 117 |
| 163 | D.3.5 | Property Definition .....                        | 120 |
| 164 | D.3.6 | CRUDN behavior .....                             | 120 |
| 165 | D.4   | Device .....                                     | 120 |
| 166 | D.4.1 | Introduction .....                               | 120 |
| 167 | D.4.2 | Wellknown URI .....                              | 120 |
| 168 | D.4.3 | Resource Type .....                              | 120 |
| 169 | D.4.4 | RAML Definition .....                            | 120 |
| 170 | D.4.5 | Property Definition .....                        | 122 |
| 171 | D.4.6 | CRUDN behavior .....                             | 122 |
| 172 | D.5   | Maintenance .....                                | 122 |
| 173 | D.5.1 | Introduction .....                               | 122 |
| 174 | D.5.2 | Wellknown URI .....                              | 122 |
| 175 | D.5.3 | Resource Type .....                              | 122 |
| 176 | D.5.4 | RAML Definition .....                            | 122 |
| 177 | D.5.5 | Property Definition .....                        | 125 |
| 178 | D.5.6 | CRUDN behavior .....                             | 125 |
| 179 | D.6   | Platform .....                                   | 125 |
| 180 | D.6.1 | Introduction .....                               | 125 |
| 181 | D.6.2 | Wellknown URI .....                              | 125 |
| 182 | D.6.3 | Resource Type .....                              | 125 |
| 183 | D.6.4 | RAML Definition .....                            | 125 |
| 184 | D.6.5 | Property Definition .....                        | 127 |
| 185 | D.6.6 | CRUDN behavior .....                             | 128 |
| 186 | D.7   | Ping .....                                       | 128 |
| 187 | D.7.1 | Introduction .....                               | 128 |
| 188 | D.7.2 | Wellknown URI .....                              | 128 |
| 189 | D.7.3 | Resource Type .....                              | 128 |
| 190 | D.7.4 | RAML Definition .....                            | 128 |
| 191 | D.7.5 | Property Definition .....                        | 129 |
| 192 | D.7.6 | CRUDN behavior .....                             | 129 |
| 193 | D.8   | Discoverable Resources, baseline interface ..... | 129 |
| 194 | D.8.1 | Introduction .....                               | 129 |
| 195 | D.8.2 | Wellknown URI .....                              | 129 |

|     |        |   |     |
|-----|--------|---|-----|
| 196 | D.8.3  | Resource Type .....                               | 129 |
| 197 | D.8.4  | RAML Definition .....                             | 129 |
| 198 | D.8.5  | Property Definition .....                         | 131 |
| 199 | D.8.6  | CRUDN behavior .....                              | 132 |
| 200 | D.9    | Discoverable Resources, link list interface ..... | 132 |
| 201 | D.9.1  | Introduction .....                                | 132 |
| 202 | D.9.2  | Wellknown URI .....                               | 132 |
| 203 | D.9.3  | Resource Type .....                               | 132 |
| 204 | D.9.4  | RAML Definition .....                             | 132 |
| 205 | D.9.5  | Property Definition .....                         | 133 |
| 206 | D.9.6  | CRUDN behavior .....                              | 134 |
| 207 | D.9.7  | Referenced JSON schemas .....                     | 135 |
| 208 | D.9.8  | oic.oic-link-schema.json .....                    | 135 |
| 209 | D.10   | Scenes (Top level) .....                          | 137 |
| 210 | D.10.1 | Introduction .....                                | 137 |
| 211 | D.10.2 | Example URI .....                                 | 137 |
| 212 | D.10.3 | Resource Type .....                               | 137 |
| 213 | D.10.4 | RAML Definition .....                             | 137 |
| 214 | D.10.5 | Property Definition .....                         | 139 |
| 215 | D.10.6 | CRUDN behavior .....                              | 139 |
| 216 | D.11   | Scene Collections .....                           | 140 |
| 217 | D.11.1 | Introduction .....                                | 140 |
| 218 | D.11.2 | Example URI .....                                 | 140 |
| 219 | D.11.3 | Resource Type .....                               | 140 |
| 220 | D.11.4 | RAML Definition .....                             | 140 |
| 221 | D.11.5 | Property Definition .....                         | 143 |
| 222 | D.11.6 | CRUDN behavior .....                              | 144 |
| 223 | D.12   | Scene Member .....                                | 144 |
| 224 | D.12.1 | Introduction .....                                | 144 |
| 225 | D.12.2 | Example URI .....                                 | 144 |
| 226 | D.12.3 | Resource Type .....                               | 144 |
| 227 | D.12.4 | RAML Definition .....                             | 144 |
| 228 | D.12.5 | Property Definition .....                         | 146 |
| 229 | D.12.6 | CRUDN behavior .....                              | 146 |
| 230 | D.13   | Resource directory resource .....                 | 146 |
| 231 | D.13.1 | Introduction .....                                | 146 |
| 232 | D.13.2 | Wellknown URI .....                               | 147 |
| 233 | D.13.3 | Resource Type .....                               | 147 |
| 234 | D.13.4 | RAML Definition .....                             | 147 |
| 235 | D.13.5 | Property Definition .....                         | 151 |
| 236 | D.13.6 | CRUDN behavior .....                              | 151 |
| 237 |        |   |     |
| 238 |        |   |     |

239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278

## Figures

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



|     |  |     |
|-----|--|-----|
| 279 | Figure 37. When at home: from smartphone turn on a single light.....                     | 99  |
| 280 | Figure 38. Device management (maintenance) .....   | 100 |
| 281 | Figure 39. Direct interaction between Server and Client .....                            | 101 |
| 282 | Figure 40. Interaction between Client and Server using another Server .....              | 101 |
| 283 | Figure 41. Interaction between Client and Server using Intermediary.....                 | 101 |
| 284 | Figure 42. Interaction between Client and Server using support from multiple Servers and |     |
| 285 | Intermediary .....   | 102 |
| 286 | Figure 43. Example of Devices .....  | 102 |
| 287 |  |     |
| 288 |  |     |
| 289 |  |     |

## Tables

|     |   |     |
|-----|---|-----|
| 290 |   |     |
| 291 |   |     |
| 292 | Table 1. Data type definition .....   | 17  |
| 293 | Table 2. Name Property Definition .....   | 30  |
| 294 | Table 3. Resource Identity Property Definition .....                                    | 30  |
| 295 | Table 4. Resource Type Common Property definition .....                                 | 31  |
| 296 | Table 5. Example foobar Resource Type .....   | 32  |
| 297 | Table 6. Example foobar properties .....  | 32  |
| 298 | Table 7. Resource Interface Property definition .....                                   | 33  |
| 299 | Table 8. OCF standard Interfaces .....  | 34  |
| 300 | Table 9: Common Properties for Collections (in addition to Common Properties defined in |     |
| 301 | section 7.3.2) .....  | 51  |
| 302 | Table 10. Parameters of CRUDN messages .....  | 52  |
| 303 | Table 11. List of Core Resources .....  | 61  |
| 304 | Table 12. Configuration Resources .....   | 65  |
| 305 | Table 13. oic.wk.con resource type definition .....                                     | 65  |
| 306 | Table 14. Mandatory discovery Core Resources .....                                      | 69  |
| 307 | Table 15. oic.wk.res resource type definition .....                                     | 70  |
| 308 | Table 16. Protocol scheme registry .....  | 70  |
| 309 | Table 17. oic.wk.d resource type definition .....                                       | 71  |
| 310 | Table 18. oic.wk.p resource type definition .....                                       | 72  |
| 311 | Table 19: Selection parameters .....  | 78  |
| 312 | Table 20. Optional diagnostics and maintenance device management Core Resources .....   | 83  |
| 313 | Table 21. oic.wk.mnt resource type definition .....                                     | 83  |
| 314 | Table 22 list of resource types for Scenes .....  | 88  |
| 315 | Table 23. CoAP request and response .....   | 89  |
| 316 | Table 24. Content Types and Content Formats .....                                       | 91  |
| 317 | Table 25. Ping resource .....   | 93  |
| 318 | Table 26. oic.wk.ping resource type definition .....                                    | 93  |
| 319 | Table 27. oic.example.light resource type definition .....                              | 98  |
| 320 | Table 28. oic.example.garagedoor resource type definition .....                         | 98  |
| 321 | Table 29. Light control resource type definition .....                                  | 106 |
| 322 | Table 30. Light control resource type definition .....                                  | 106 |
| 323 | Table 31. Alphabetized list of core resources .....                                     | 108 |
| 324 |   |     |
| 325 |   |     |

## 326 1 Scope

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

- 328 • Core Specification documents: The Core Specification documents specify the Framework, i.e.,  
329 the OCF core architecture, interfaces, protocols and services to enable OCF profiles  
330 implementation for Internet of Things (IoT) usages and ecosystems.
- 331 • Vertical Profiles Specification documents: The Vertical Profiles Specification documents  
332 specify the OCF profiles to enable IoT usages for different market segments such as smart  
333 home, industrial, healthcare, and automotive. The Application Profiles Specification is built  
334 upon the interfaces and network security of the OCF core architecture defined in the Core  
335 Specification.

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

337

## 338 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

419

420 **3 Terms, definitions, symbols and abbreviations**

421 **3.1 Terms and definitions**

422 **3.1.1**

423 **Client**

424 a logical entity that accesses a Resource on a Server

425 **3.1.2**

426 **Collection**

427 a Resource that contains zero or more Links

428 **3.1.3**

429 **Configuration Source**

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

432 **3.1.4**

433 **Core Resources**

434 those Resources that are defined in this specification

435 **3.1.5**

436 **Default Interface**

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

438 **3.1.6**

439 **Device**

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

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

442 **3.1.7**

443 **Device Type**

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

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

447 **3.1.8**

448 **Entity**

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

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

451 **3.1.9**

452 **Framework**

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

455 **3.1.10**

456 **Links**

457 extends typed web links as specified in IETF RFC 5988

458 **3.1.11**

459 **Non-OCF Device**

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

461 **3.1.12**

462 **Notification**

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

464 **3.1.13**  
465 **Observe**  
466 the act of monitoring a Resource by sending a RETRIEVE request which is cached by the Server  
467 hosting the Resource and reprocessed on every change to that Resource

468 **3.1.14**  
469 **Parameter**  
470 an element that provides metadata about a Resource referenced by the target URI of a Link

471 **3.1.15**  
472 **Partial UPDATE**  
473 an UPDATE request to a Resource that includes a subset of the Properties that are visible via the  
474 Interface being applied for the Resource Type

475 **3.1.16**  
476 **Platform**  
477 a physical device containing one or more Devices

478 **3.1.17**  
479 **Remote Access Endpoint (RAE) Client**  
480 a Client which supports XMPP functionality in order to access a Server from a remote location

481 **3.1.18**  
482 **Remote Access Endpoint (RAE) Server**  
483 a Server which supports XMPP and can publish its resource(s) to an XMPP server in the Cloud,  
484 thus becoming remotely addressable and accessible

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

486 **3.1.19**  
487 **Resource**  
488 represents an Entity modelled and exposed by the Framework

489 **3.1.20**  
490 **Resource Directory**  
491 a set of descriptions of resources where the actual resources are held on Servers external to the  
492 Device hosting the Resource Directory, allowing lookups to be performed for those resources

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

495 **3.1.21**  
496 **Resource Interface**  
497 a qualification of the permitted requests on a Resource

498 **3.1.22**  
499 **Resource Property**  
500 a significant aspect or parameter of a resource, including metadata, that is exposed through the  
501 Resource

502 **3.1.23**  
503 **Resource Type**  
504 a uniquely named definition of a class of Resource Properties and the interactions that are  
505 supported by that class

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

507 **3.1.24**  
508 **Scene**  
509 a static entity that stores a set of defined Resource property values for a collection of Resources

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

### 512 **3.1.25**

#### 513 **Scene Collection**

514 a collection Resource that contains an enumeration of possible Scene Values and the current  
515 Scene Value

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

### 517 **3.1.26**

#### 518 **Scene Member**

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

### 520 **3.1.27**

#### 521 **Scene Value**

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

### 523 **3.1.28**

#### 524 **Server**

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

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

## 528 **3.2 Symbols and abbreviations**

### 529 **3.2.1**

#### 530 **ACL**

531 Access Control List

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

### 533 **3.2.2**

#### 534 **CBOR**

535 Concise Binary Object Representation

### 536 **3.2.3**

#### 537 **CoAP**

538 Constrained Application Protocol

### 539 **3.2.4**

#### 540 **EXI**

541 Efficient XML Interchange

### 542 **3.2.5**

#### 543 **IRI**

544 Internationalized Resource Identifiers

### 545 **3.2.6**

#### 546 **ISP**

547 Internet Service Provider

### 548 **3.2.7**

#### 549 **JSON**

550 JavaScript Object Notation

### 551 **3.2.8**

#### 552 **mDNS**

553 Multicast Domain Name Service



554 **3.2.9**  
 555 **MTU**  
 556 Maximum Transmission Unit

557 **3.2.10**  
 558 **NAT**  
 559 Network Address Translation

560 **3.2.11**  
 561 **OCF**  
 562 Open Connectivity Foundation

563 the organization that created this specification

564 **3.2.12**  
 565 **URI**  
 566 Uniform Resource Identifier

567 **3.2.13**  
 568 **URN**  
 569 Uniform Resource Name

570 **3.2.14**  
 571 **UTC**  
 572 Coordinated Universal Time

573 **3.2.15**  
 574 **UUID**  
 575 Universal Unique Identifier

576 **3.2.16**  
 577 **XML**  
 578 Extensible Markup Language

579 **3.3 Conventions**

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

584 **3.4 Data types**

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

590 **Table 1. Data type definition**

| Name           | JSON value | JSON format value | Description   |
|----------------|------------|-------------------|---|
| <b>boolean</b> | false true | n/a               | Binary-value {0, 1}.  |
| <b>BSV</b>     | string     | bsv               | A blank (i.e. space) separated list of values encoded within a string. The value type in the BSV is described by the property where the BSV is used. For example a BSV of integers. |

|                 |              |           |  |
|-----------------|--------------|-----------|--|
| <b>CSV</b>      | string       | csv       | A comma separated list of values encoded within a string. The value type in the CSV is described by the property where the CSV is used. For example a CSV of integers.                   |
| <b>date</b>     | string       | date-time | As defined in ISO 8601. The format is restricted to [yyyy]-[mm]-[dd].  |
| <b>datetime</b> | string       | date-time | As defined in ISO 8601.  |
| <b>enum</b>     | enum         | n/a       | Enumerated type.   |
| <b>float</b>    | number       | float     | Signed IEEE 754 single precision float value.  |
| <b>integer</b>  | number       | integer   | Signed 32 bit integer.   |
| <b>json</b>     | object/array | n/a       | A data represented using a JSON element which could be an object or array as defined in ECMA-4-4. The JSON object or array needs to be described by means of a JSON schema.              |
| <b>string</b>   | string       | n/a       | UTF-8 character string shall not exceed a max length of 64 octets unless otherwise specified for a Property value in this specification.   |
| <b>time</b>     | string       | time      | As defined in ISO 8601 but restricted to UTC with a trailing "Z". The format is [hh]:[mm]:[ss]Z.   |
| <b>URI</b>      | string       | uri       | A uniform resource identifier (URI) is a string of characters used to identify a resource according to IETF RFC 3986. The URI value shall not exceed a max length of 256 octets (bytes). |
| <b>UUID</b>     | string       | uuid      | An identifier formatted according to IETF RFC 4122.  |

591

#### 592 **4 Document conventions and organization**

593 In this document, features are described as required, recommended, allowed or DEPRECATED as  
594 follows:

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

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

599 Recommended (or should)(S).

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

606 Allowed (may or allowed)(O).

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

610 DEPRECATED.

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

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

617 Conditionally allowed (CA)

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

620 Conditionally required (CR)

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

624

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

626 Words that are emphasized are printed in italic.

## 627 **5 Architecture**

### 628 **5.1 Overview**

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

633 Specifically, the architecture provides:

- 634 • A communication and interoperability framework for multiple market segments (Consumer,  
635 Enterprise, Industrial, Automotive, Health, etc.), OSs, platforms, modes of communication,  
636 transports and use cases
- 637 • A common and consistent model for describing the environment and enabling information  
638 and semantic interoperability
- 639 • Common communication protocols for discovery and connectivity
- 640 • Common security and identification mechanisms
- 641 • Opportunity for innovation and product differentiation
- 642 • A scalable solution addressing different device capabilities, applicable to smart devices as  
643 well as the smallest connected things and wearable devices

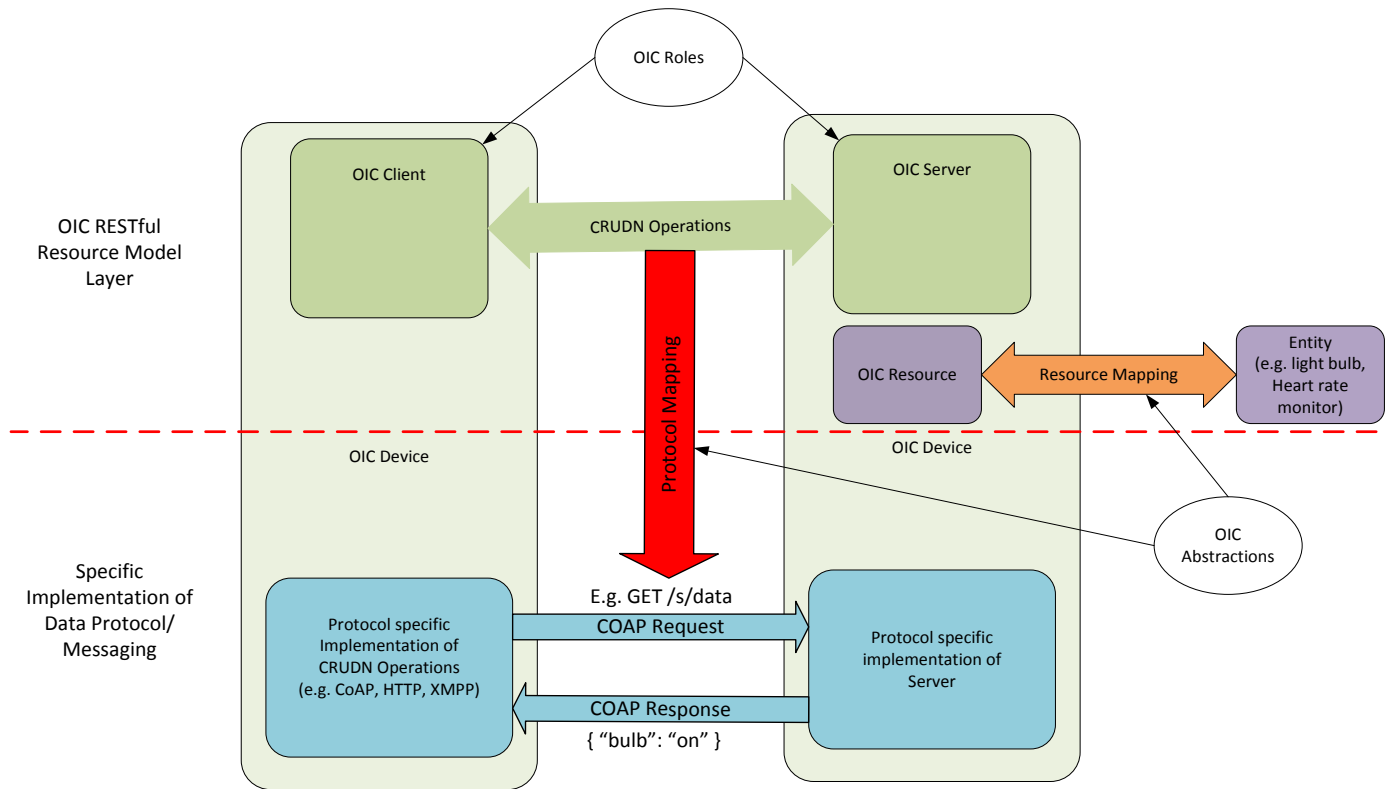
644 The architecture is based on the Resource Oriented Architecture design principles and described  
645 in the sections 5.2 through 5.5 respectively. Section 5.2 presents the guiding principles for OCF  
646 operations. Section 5.3 defines the functional block diagram and Framework. Section 5.4 provides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

### 649 **5.2 Principle**

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

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

665 Figure 1 depicts the architecture.



666  
667

668 **Figure 1: Architecture - concepts**

669

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

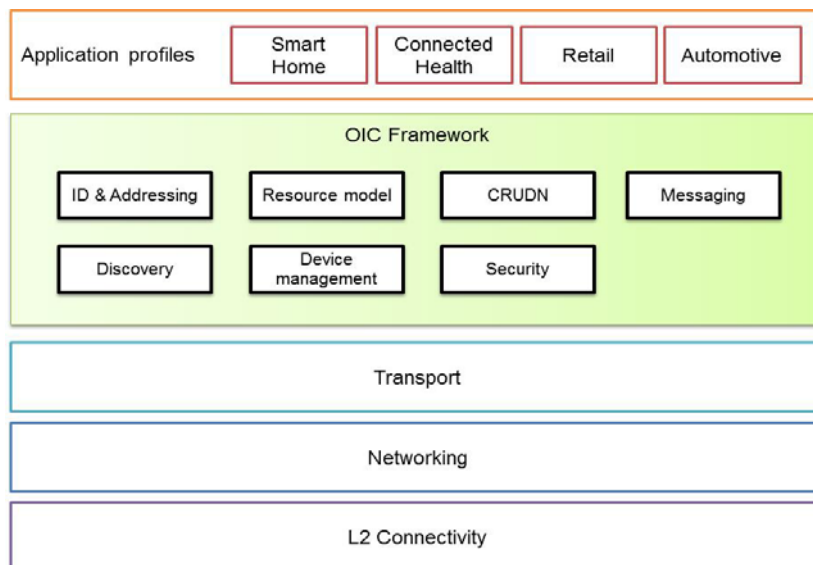
- 672 • Resource model: The resource model provides the abstractions and concepts required to  
 673 logically model, and logically operate on the application and its environment. The core resource  
 674 model is common and agnostic to any specific application domain such as smart home,  
 675 industrial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

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

688

### 689 5.3 Functional block diagram

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



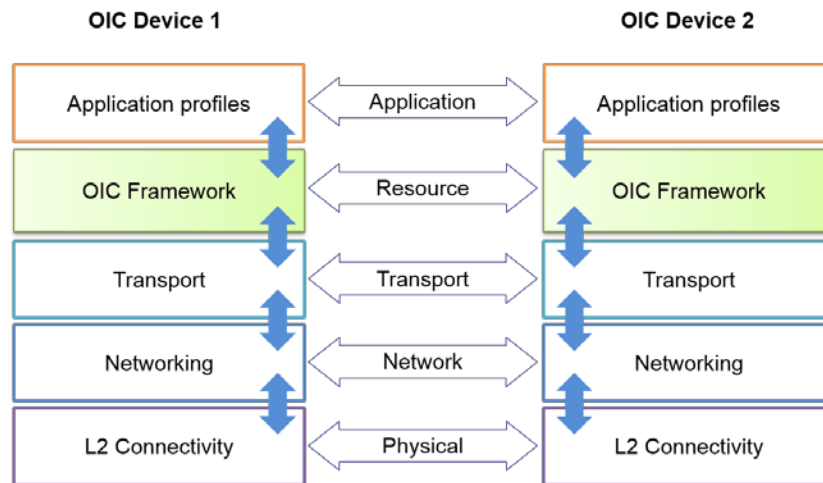
693

694

**Figure 2: Functional block diagram**

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

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



709  
 710 **Figure 3: Communication layering model**

711 **5.3.1 Framework**

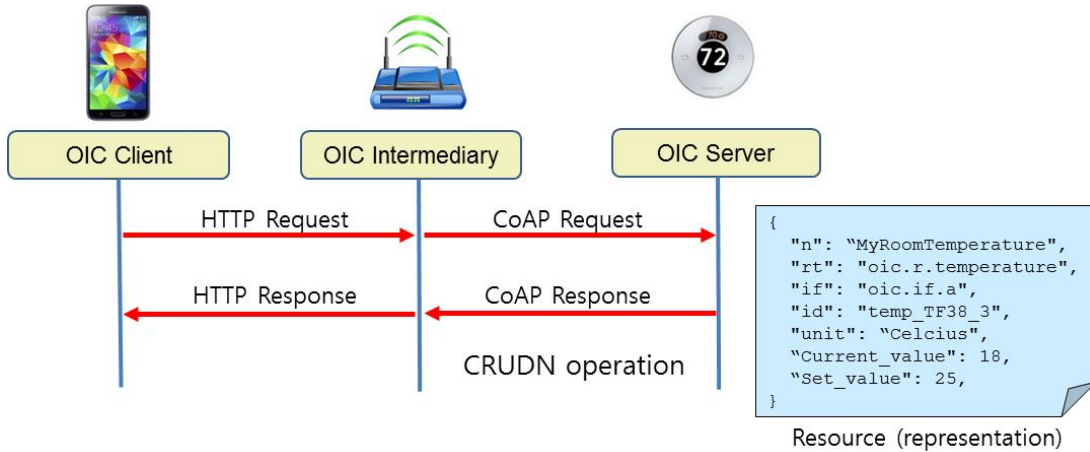
712 Framework consists of functions which provide core functionalities for operation.

- 713 1) **Identification and addressing.** Defines the identifier and addressing capability. The  
 714 Identification and addressing function is defined in section 6.
- 715 2) **Discovery.** Defines the process for discovering available  
 716 a) Devices (Endpoint Discovery in section 10) and  
 717 b) Resources (Resource discovery in section 11.3)
- 718 3) **Resource model.** Specifies the capability for representation of Entities in terms of resources  
 719 and defines mechanisms for manipulating the resources. The resource model function is  
 720 defined in section 7.
- 721 4) **CRUDN.** Provides a generic scheme for the interactions between a Client and Server as  
 722 defined in section 8.
- 723 5) **Messaging.** Provides specific message protocols for RESTful operation, i.e. CRUDN. For  
 724 example, CoAP is a primary messaging protocol. The messaging function is defined in section  
 725 12.
- 726 6) **Device management.** Specifies the discipline of managing the capabilities of a Device, and  
 727 includes device provisioning and initial setup as well as device monitoring and diagnostics.  
 728 The device management function is defined in section 11.5.
- 729 7) **Security.** Includes authentication, authorization, and access control mechanisms required for  
 730 secure access to Entities. The security function is defined in section 13.

731 **5.4 Example Scenario with roles**

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

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



739

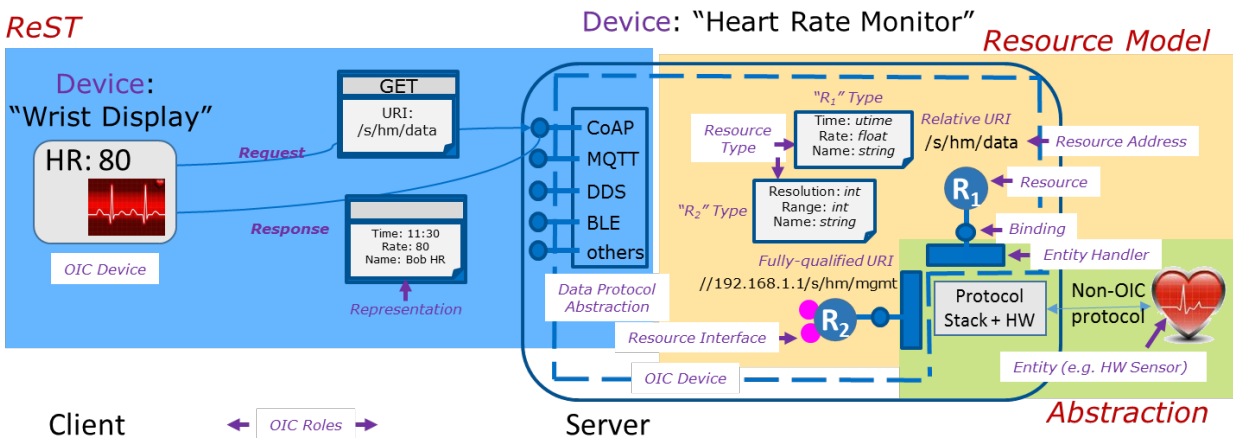
740

**Figure 4: Example illustrating the Roles**

741 **5.5 Example Scenario: Bridging to Non- OCF ecosystem**

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

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



745

746

**Figure 5: Framework - Architecture Detail**

747

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



750  
751

752

**Figure 6: Server bridging to Non- OCF device**

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

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

## 762 **6 Identification and addressing**

### 763 **6.1 Introduction**

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

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

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

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

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

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

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

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



## 787 6.2 Identification

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

### 802 6.2.1 Resource identification and addressing

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

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

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

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

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

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

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

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

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

820 The usual form of the authority is

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

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

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

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

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

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

840 A URI may be either

- 841 • Fully qualified or
- 842 • Relative

843 *Generation of URI:*

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

849 *Use of URI:*

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

### 855 **6.3 Namespace:**

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

### 858 **6.4 Network addressing**

859 The following are the addresses used in this specification:

- 860 • **IP address**

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

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

## 864 **7 Resource model**

### 865 **7.1 Introduction**

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

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

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

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

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

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

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

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

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

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

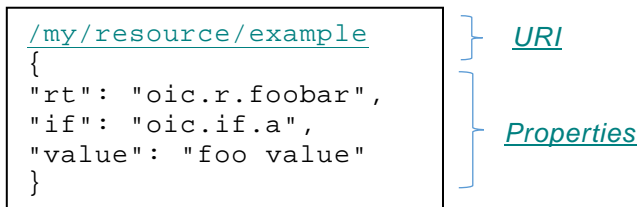
903 The RAML definitions used in this document are normative. This also includes that all defined  
904 JSON payloads shall comply with the indicated JSON schema. See Annex D for Resource Types  
905 defined in this specification.

## 906 **7.2 Resource**

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

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

912 specification of the Resource Type.



913

914

**Figure 7: Example of a Resource**

915

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

## 920 7.3 Property

### 921 7.3.1 Introduction

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

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

930 In addition, the Property definition shall have a

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

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

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

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

958 In general, a property is meaningful only within the resource to which it is associated. However a  
959 base set of properties that may be supported by all Resources, known as Common Properties,  
960 keep their semantics intact across Resources i.e. their “key=value” pair means the same in any  
961 Resource. Detailed tables with the above fields for all common properties are defined in section  
962 7.3.2.

## 963 **7.3.2 Common Properties**

### 964 **7.3.2.1 Introduction**

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

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

980 The following Common Properties for all Resources are specified in section 7.3.2.2 through section  
981 7.3.2.6 and summarized as follows:

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

996 **7.3.2.2 Property Name and Property Value definitions**

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

- 998 • **Property Name**– the key in "key=value" pair. Property Name is case sensitive and its data type  
999 is "string" but only ASCII characters are permitted, and embedded NUL characters are not  
1000 permitted.
- 1001 • **Property Value** – the value in "key=value" pair. Property Value is case sensitive when its data  
1002 type is "string". Any enum values shall be ASCII only.

1003 **7.3.2.3 Resource Type**

1004 Resource Type Property is specified in Section 7.4.

1005 **7.3.2.4 Interface**

1006 Interface Property is specified in Section 7.5.

1007 **7.3.2.5 Name**

1008 A human friendly name for the resource, i.e. a specific resource instance name (e.g.,  
1009 MyLivingRoomLight), The Name Property is as defined in Table 2

1010 **Table 2. Name Property Definition**

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

1011

1012 **7.3.2.6 Resource Identity**

1013 The Resource Identity Property shall be a unique (across the scope of the host Server) instance  
1014 identifier for a specific instance of the Resource. The encoding of this identifier is device and  
1015 implementation dependent. The Resource Identity Property is as defined in Table 3.

1016 **Table 3. Resource Identity Property Definition**

1017

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

1018

1019 **7.4 Resource Type**

1020 **7.4.1 Introduction**

1021 Resource Type is a class or category of Resources and a Resource is an instance of one or more  
1022 Resource Types.

1023 The Resource Types of a Resource is declared using the Resource Type Common Property as  
1024 described in Section 7.3.2.3 or in a Link using the Resource Type Parameter.

1025 A Resource Type may either be pre-defined (Core Resource Types in this specification and vertical  
 1026 Resource Types in vertical domain specifications) or in custom definitions by manufacturers, end  
 1027 users, or developers of Devices (vendor-defined Resource Types). Resource Types and their  
 1028 definition details may be communicated out of band (like in documentation) or be defined explicitly  
 1029 using a meta-language which may be downloaded and used by APIs or applications. OCF has  
 1030 adopted RAML and JSON Schema as the specification method for OCF's RESTful interfaces and  
 1031 Resource definitions. OCF defined Interfaces and Resource Types are specified using RAML and  
 1032 JSON schema (respectively).

1033 Every Resource Type shall be identified with a Resource Type ID which shall be a lower case  
 1034 string with segments separated by a "." (dot). The entire string represents the Resource Type ID.  
 1035 When defining the ID each segment may represent any semantics that are appropriate to the  
 1036 Resource Type. For example, each segment could represent a namespace. Once the ID has been  
 1037 defined, the ID should be used opaquely and an implementations should not infer any information  
 1038 from the individual segments. The string "oic", when used as the first segment in the definition  
 1039 of the Resource Type ID, is reserved for OCF-defined Resource Types. The Resource Type ID  
 1040 may also be a reference to an authority similar to IANA that may be used to find the definition of a  
 1041 Resource Type.

#### 1042 7.4.2 Resource Type Property

1043 A Resource when instantiated or created shall have one or more Resource Types that are the  
 1044 template for that Resource. The Resource Types that the Resource conforms to shall be declared  
 1045 using the "rt" Common Property for the Resource. The Property Value for the "rt" Common Property  
 1046 shall be the list of Resource Type IDs for the Resource Types used as templates (i.e., "rt"=<list of  
 1047 Resource Type IDs>).

1048 **Table 4. Resource Type Common Property definition**

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

1049 Resource Types may be explicitly discovered or implicitly shared between the user (i.e. Client) and  
 1050 the host (i.e. Server) of the Resource.

#### 1051 7.4.3 Resource Type definition

1052 Resource Type is specified as follows:

- 1053 • **Pre-defined URI (optional)** – a pre-defined URI may be specified for a specific Resource Type  
 1054 in an OCF specification. When a Resource Type has a pre-defined URI, all instances of that  
 1055 Resource Type shall use only the pre-defined URI. An instance of a different Resource Type  
 1056 shall not use the pre-defined URI.
- 1057 • **Resource Type Title (optional)** – a human friendly name to designate the resource type.
- 1058 • **Resource Type ID** – the value of "rt" property which identifies the Resource Type, (e.g.,  
 1059 oic.wk.p). A lower case string that has segments separated by a '.' (dot); each segment may  
 1060 represent a name space and in that case later segments (L -> R) would represent sub-name  
 1061 spaces; Implementations shall use these opaquely and use case sensitive string matches.
- 1062 • **Resource Interfaces** – list of the interfaces that may be supported by the resource type.
- 1063 • **Resource Properties** – definition of all the properties that apply to the resource type. The  
 1064 resource type definition shall define whether a property is mandatory, conditional mandatory,  
 1065 or optional.

- **Related Resource Types** (optional) – the specification 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 provide 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) | 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      |            |      | R           | yes       | Resource type |
| Interface      | if            | array      |            |      | R           | yes       | Interface     |
| Foo value      | value         | string     |            |      | R           | yes       | Foo value     |

An instance of the foobar resource type is as shown below

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

An example schema for the foobar resource type is shown below

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

## 7.5 Device Type

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



1091 additional standard and vendor defined Resource Types beyond the minimum list. The Device  
1092 Type is used in Resource discovery as specified in section 11.3.4.

1093 Like a Resource Type, a Device Type can be used in the Resource Type Common Property or in  
1094 a Link using the Resource Type Parameter.

1095 A Device Type may either be pre-defined (in vertical domain specifications) or in custom definitions  
1096 by manufacturers, end users, or developers of Devices (vendor-defined Device Types). Device  
1097 Types and their definition details may be communicated out of band (like in documentation).

1098 Every Device Type shall be identified with a Resource Type ID using the same syntax constraints  
1099 as a Resource Type.

## 1100 **7.6 Interface**

### 1101 **7.6.1 Introduction**

1102 An Interface provides first a view into the Resource and then defines the requests and responses  
1103 permissible on that view of the Resource. So this view provided by an Interface defines the context  
1104 for requests and responses on a Resource. Therefore, the same request to a Resource when  
1105 targeted to different Interfaces may result in different responses.

1106 An Interface may be defined by either this specification (a Core Interface), the OCF vertical domain  
1107 specifications (a “vertical Interface) or manufacturers, end users or developers of Devices (a  
1108 “vendor-defined Interface”).

1109 The Interface Property lists all the Interfaces the Resource support. All resources shall have at  
1110 least one Interface. The Default Interface shall be defined by an OCF specification and inherited  
1111 from the resource type definition. The Default Interface associated with all Resource Types defined  
1112 in this specification shall be the supported Interface listed first within the applicable enumeration  
1113 in the definition of the Resource Type (see Annex D). All Default Interfaces specified in an OCF  
1114 specification shall be mandatory.

1115 In addition to any OCF specification defined interface, all Resources shall support the Baseline  
1116 Interface (oic.if.baseline) as defined in section 7.6.3.2.

1117 When an Interface is to be selected for a Request, it shall be specified as query parameter in the  
1118 URI of the Resource in the Request message. If no query parameter is specified, then the Default  
1119 Interface shall be used. If the selected Interface is not one of the permitted Interfaces on the  
1120 Resource then selecting that Interface is an error.

1121 An Interface may accept more than one media type. An Interface may respond with more than one  
1122 media type. The accepted media types may be different from the response media types. The media  
1123 types are specified with the appropriate header parameters in the transfer protocol. (NOTE: This  
1124 feature has to be used judiciously and is allowed to optimize representations on the wire) Each  
1125 Interface shall have at least one media type.

1126

### 1127 **7.6.2 Interface Property**

1128 **Table 7. Resource Interface Property definition**

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

1129 The Interfaces supported by a Resource shall be declared using the Interface Common Property  
 1130 (Table 7) as "if=<array of Interfaces>". The Property Value of an Interface Property shall be a  
 1131 lower case string with segments separated by a "." (dot). The string "oic", when used as the first  
 1132 segment in the Interface Property Value, is reserved for OCF-defined Interfaces. The Interface  
 1133 Property Value may also be a reference to an authority similar to IANA that may be used to find  
 1134 the definition of an Interface. A Resource Type shall support one or more of the Interfaces defined  
 1135 in section 7.6.3.

1136 **7.6.3 Interface methods**

1137 **7.6.3.1 Overview**

1138 The OCF -defined Interfaces are listed in the table below:

1139 **Table 8. OCF standard Interfaces**

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

1140

1141

1142 **7.6.3.2 Baseline Interface**

1143 **7.6.3.2.1 Overview**

1144 The Representation that is visible using the "baseline" Interface includes all the Properties of the  
 1145 Resource including the Common Properties. The "baseline" Interface shall be defined for all  
 1146 Resource Types. All Resources shall support the "baseline" Interface.

1147 The "baseline" Interface is selected by adding if=oic.if.baseline to the list of query parameters in  
 1148 the URI of the target Resource. For example: GET /oic/res?if=oic.if.baseline.

1149 **7.6.3.2.2 Use of RETRIEVE**

1150 The “baseline” Interface is used when a Client wants to retrieve all Properties of a Resource. The  
1151 Client includes the URI query parameter definition “?if=oic.if.baseline” in a RETRIEVE request.  
1152 When this query parameter definition is included the Server shall respond with a Resource  
1153 representation that includes all of the implemented Properties of the Resource. When the Server  
1154 is unable to send back the whole Resource representation, it shall reply with an error message.  
1155 The Server shall not return a partial Resource representation.

1156 An example response to a RETRIEVE request using the baseline Interface is shown below:

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

1157

1158 **7.6.3.2.3 Use of UPDATE**

1159 Using the baseline Interface, all Properties of a Resource may be modified using an UPDATE  
1160 request with a list of Properties and their desired values.

1161 **7.6.3.3 Link List Interface**

1162 **7.6.3.3.1 Overview**

1163 The links list Interface provides a view into the list of Links in a Collection (Resource). The  
1164 Representation visible through this Interface has only the Links defined in the Property Value of  
1165 the “links” Property – so this Interface is used to manipulate or interact with the list of Links in a  
1166 Collection. The Links list may be RETRIEVED using this Interface.

1167 The Interface definition and semantics are given as follows:

- 1168 • The links list Interface name shall be “oic.if.ll”.
- 1169 • If specified in a request (usually in the request header), the serialization in the response shall  
1170 be in the format expected in the request.
- 1171 • In response to a RETRIEVE request on the “links list” Interface, the URIs of the referenced  
1172 Resources shall be returned as a URI reference.
- 1173 • If there are no links present in a Resource, then an empty list shall be returned.
- 1174 • The Representation determined by this Interface view only includes the Property Value of the  
1175 “links” Property.

1176 **7.6.3.3.2 Example: “links list” Interface**

1177 **Example: Request to a Collection**

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

1178

### 1179 7.6.3.4 Batch Interface

#### 1180 7.6.3.4.1 Overview

1181 The batch Interface is used to interact with a collection of Resources using a single/same Request.  
1182 The batch Interface supports methods of Resources in the Links of the Collection, and can be used  
1183 to RETRIEVE or UPDATE the Properties of the “linked” Resources with a single Resource  
1184 representation.

1185 The batch Interface selects a view into the Links in a Collection – the Request is sent to all the  
1186 Links in this view with potential modifications defined in the Parameters of the Link

1187 The batch Interface is defined as follows:

- 1188 • The batch Interface name shall be “oic.if.b”
- 1189 • A Resource with a batch Interface has Links that have Resource references that may be URIs  
1190 (fully qualified for remote Resources) or relative references (for local Resources).
- 1191 • If the Link to a Resource does not specify an Interface to use (using the “bp” Link parameter),  
1192 then the Request shall be forwarded to the Default Interface of the referenced Resource. If the  
1193 “bp” specifies a query using the “q” key then that query shall be used in the query parameter  
1194 of the URI formed from the Reference so as to select that Interface in the target Resource.  
1195 (See “Link” section for more information on “bp” Parameter)
- 1196 • The original request is modified to create new requests targeting each of the targets in the  
1197 Resource Links by substituting the URI in the original request with the URI of the target  
1198 Resource in the Link. The payload in the original request is replicated in the payload of the  
1199 new Requests.
- 1200 • All the Responses from the “linked” Resources shall be aggregated into single Response to  
1201 the Client. The Server may timeout the Response to a time window (if a time window has been  
1202 negotiated with the Client then the Server shall not timeout within that window; in the absence  
1203 of negotiated window, the Server may choose any appropriate window based on conditions). If  
1204 the target Resources cannot process the new request, an empty response or error response  
1205 shall be returned. These empty/error Responses shall be included in aggregated Response to  
1206 the original Client Request.
- 1207 • The aggregate Response is an array of objects with individual responses. Each response in  
1208 the aggregate shall include at least two items: (1) the URI (fully qualified) as “href”: <URI> and  
1209 (2) the Representation in the Response declared using the keyword “rep” as the key i.e. “rep”:  
1210 { <Representation in individual Response> }.
- 1211 • The Client may choose to restrict the list of Links to which the Request is forwarded by providing  
1212 a “filter” in the URI of the Collection to which this original ‘batch’ Interface Request is made.
- 1213 • The Representation in the Link-specific Request may not match the Representation from the  
1214 view exposed by the Interface on the target Resource. In such cases, UPDATE using ‘PUT’  
1215 method will usually fail and so UPDATE using ‘POST’ method would be appropriate – in this  
1216 case the ‘subset’ semantics apply where Properties in the Request which match Properties in  
1217 the Resource view exposed shall be modified in the target Resource if the Property is writeable.
- 1218 • A Device that supports the ‘batch’ Interface shall implement both the Client and Server Roles.

#### 1219 7.6.3.4.2 Examples: Batch Interface

1220 Example 1

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

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

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

1221

1222 Example that further shows the "links list" and "batch" interface

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

1223

1224 **7.6.3.5 Actuator Interface**

1225 The actuator Interface is the Interface for viewing Resources that may be actuated i.e. changes  
1226 some value within or the state of the entity abstracted by the Resource:

- 1227 • The actuator Interface name shall be “oic.if.a”
- 1228 • The actuator Interface shall expose in the Resource Representation all mandatory Properties  
1229 as defined by the applicable JSON; the actuator interface may also expose in the Resource  
1230 Representation optional Properties as defined by the applicable JSON schema that are  
1231 implemented by the target Device.

For the following Resource

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

```

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

```

```
}

```

1232

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

1233

**NOTE: The example here is with respect to Figure 8**

1. Retrieving values of an actuator

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

Response:

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

2. Correct use of actuator:

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

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

Response:

```
{
  Ok
}
```

3. Incorrect use of actuator

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

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

Interface

```
{
}
```

Response:

```
{
  Error
}
```

1234

**Figure 9: Example - Actuator Interface**

1235

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

1236

1237

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

1238

### 1239 7.6.3.6 Sensor Interface

1240

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

1241

1242

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

1243

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

1244



- 1245 Representation optional Properties as defined by the applicable JSON schema that are  
1246 implemented by the target Device.
- 1247 • A RETRIEVE request using this Interface shall return this Representation for the Resource  
1248 subject to any query and filter parameters that may also exist
  - 1249 •

**NOTE: The example here is with respect to Figure 8**

1. Retrieving values of sensor

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

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

2. Incorrect use of sensor

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

3. Incorrect use of sensor

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

1250

### 1251 7.6.3.7 Read-only Interface

1252 The read-only Interface exposes only the Properties that may be “read”. This includes Properties  
1253 that may be “read-only”, “read-write” but not Properties that are “write-only” or “set-only”. The  
1254 applicable methods that can be applied to a Resource is RETRIEVE only. An attempt by a Client  
1255 to apply a method other than RETRIEVE to a Resource shall be rejected with an error response  
1256 code.

### 1257 7.6.3.8 Read-write Interface

1258 The read-write Interface exposes only the Properties that may be “read” and “written”. The “read-  
1259 only” Properties shall not be included in Representation for the “read-write” Interface. This is a  
1260 generic Interface to support “reading” and “setting” Properties in a Resource. The applicable  
1261 methods that can be applied to a Resource are RETRIEVE and UPDATE only. An attempt by a

1262 Client to apply a method other than RETRIEVE or UPDATE to a Resource shall be rejected with  
1263 an error response code.

## 1264 **7.7 Resource representation**

1265 Resource representation captures the state of a Resource at a particular time. The resource  
1266 representation is exchanged in the request and response interactions with a Resource. A Resource  
1267 representation may be used to retrieve or update the state of a resource.

1268 The resource representation shall not be manipulated by the data connectivity protocols and  
1269 technologies (e.g., CoAP, UDP/IP or BLE).

## 1270 **7.8 Structure**

### 1271 **7.8.1 Introduction**

1272 In many scenarios and contexts, the Resources may have either an implicit or explicit structure  
1273 between them. A structure can, for example, be a tree, a mesh, a fan-out or a fan-in. The  
1274 Framework provides the means to model and map these structures and the relationships among  
1275 Resources. The primary building block for resource structures in Framework is the collection. A  
1276 collection represents a container, which is extensible to model complex structures.

### 1277 **7.8.2 Resource Relationships**

1278 Resource relationships are expressed as Links. A Link embraces and extends typed web links  
1279 concept as a means of expressing relationships between Resources. A Link consists of a set of  
1280 Parameters that define:

- 1281 • a context URI,
- 1282 • a target URI,
- 1283 • a relation from the context URI to the target URI
- 1284 • elements that provide metadata about the target URI, the relationship or the context of the Link.

1285 The target URI is mandatory and the other items in a Link are optional. Additional items in the Link  
1286 may be made mandatory based on the use of the links in different contexts (e.g. in collections, in  
1287 discovery, in bridging etc.). Schema for the Link payload is provided in Annex D.

1288 An example of a Link is shown in

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

1289 **Figure 10: Example of a Link**

1290 Two Links are distinct from each other when at least one parameter is different. For example the  
1291 two Links shown in Figure 11 are distinct and can appear in the same list of Links.

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

1292 **Figure 11: Example of distinct Links**

1293 The specification may mandate Parameters and Parameter values as required for certain  
1294 capabilities. For all Links returned in a response to a RETRIEVE on /oic/res, if a Link does not  
1295 explicitly include the “rel” Parameter, a value of “rel”=“hosts” shall be assumed . The relation value  
1296 of “hosts” is defined by IETF RFC 6690 and registered in the IANA Registry for Link Relations at  
1297 [\[http://www.iana.org/assignments/link-relations/link-relations.xhtml\]](http://www.iana.org/assignments/link-relations/link-relations.xhtml)

1298 As shown in D.2.8 the relation between the context URI and target URI in a Link is specified using  
1299 the “rel” JSON element and the value of this element specifies the particular relation.

1300 The context URI of the Link shall implicitly be the URI of the Resource (or specifically a Collection)  
1301 that contains the Link unless the Link specifies the anchor parameter. The anchor parameter is  
1302 used to change the context URI of a Link – the relationship with the target URI is based off the  
1303 anchor URI when the anchor is specified. An example of using anchors in the context of Collections  
1304 – a floor has rooms and rooms have lights – the lights may be defined in floor as Links but the  
1305 Links will have the anchor set to the URI of the rooms that contain the lights (the relation is  
1306 contains). This allows all lights in a floor to be turned on or off together while still having the lights  
1307 defined with respect to the rooms that contain them (lights may also be turned on by using the  
1308 room URI too).

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

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

1309 **Figure 12: Example of use of anchor in Link**

1310 **7.8.2.1 Parameters**

1311 **7.8.2.1.1 “ins” or Link Instance Parameter**

1312 The “ins” parameter identifies a particular Link instance in a list of Links. The "ins" parameter may  
1313 be used to modify or delete a specific Link in a list of Links. The value of the “ins” parameter is set  
1314 at instantiation of the Link by the OCF Device (Server) that is hosting the list of Links – once it has  
1315 been set, the “ins” parameter shall not be modified for as long as the Link is a member of that list.

1316 **7.8.2.1.2 “p” or Policy Parameter**

1317 The Policy Parameter defines various rules for correctly accessing a Resource referenced by a  
1318 target URI. The Policy rules are configured by a set of key-value pairs as defined below.

1319 The policy Parameter "p" is defined by:

- 1320 • "bm" key: The "bm" key corresponds to an integer value that is interpreted as an 8-bit bitmask.  
 1321 Each bit in the bitmask corresponds to a specific Policy rule. The following rules are specified  
 1322 for "bm":  
 1323

| Bit Position                | Policy rule  | Comment   |
|-----------------------------|--------------|---|
| Bit 0 (the LSB)             | discoverable | <p>The discoverable rule defines whether the Link is to be included in the Resource discovery message via /oic/res.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul> |
| 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.</p> <ul style="list-style-type: none"> <li>• If the Resource supports the NOTIFY operation, then "p" shall include the "bm" key and set the observable bit to value 1.</li> <li>• 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.</li> </ul>   |
| Bits 2-7                    | --           | Reserved for future use. All reserved bits in "bm" shall be set to value 0.   |

1324  
 1325 Note that if all the bits in "bm" are defined to value 0, then the "bm" key may be omitted entirely  
 1326 from "p" as an efficiency measure. However, if any bit is set to value 1, then "bm" shall be  
 1327 included in "p" and all the bits shall be defined appropriately.

- 1328 • "sec" key: The "sec" key corresponds to a Boolean value that indicates whether the Resource  
 1329 referenced by the target URI is accessed via an encrypted connection. If "sec" is true, the  
 1330 resource is accessed via an encrypted connection, using the "port" specified (see below). If  
 1331 "sec" is false, the resource is accessed via an unencrypted connection, or via an encrypted  
 1332 connection (if such a connection is made using the "port" settings for another Resource, for  
 1333 which "sec" is true).
- 1334 • "port" key: The "port" key corresponds to an integer value that is used to indicate the port  
 1335 number where the Resource referenced by the target URI may be accessed via an encrypted  
 1336 connection.
- 1337 • If the Resource is only available via an encrypted connection (i.e. DTLS over IP), then
- 1338 ○ "p" shall include the "sec" key and its value shall be true.
  - 1339 ○ "p" shall include the "port" key and its value shall be the port number where the  
 1340 encrypted connection may be established.

- 1341 • If the Resource is not available via an encrypted connection, then
  - 1342 ○ "p" shall include the "sec" key and its value shall be false or "p" shall omit the "sec"
  - 1343 key; the default value of "sec" is false.
  - 1344 ○ "p" shall omit the "port" key.
  - 1345 ○ A Resource that is available via either an encrypted or unencrypted connection
  - 1346 follows the population scheme defined in this clause.
- 1347 • Access to the Resource on the port specified by the "port" key shall be made by an encrypted
- 1348 connection (e.g. coaps://). (Note that unencrypted connection to the Resource may be possible
- 1349 on a separate port discovered thru multicast discovery).
- 1350 • Note that access to the Resource is controlled by the ACL for the Resource. A successful
- 1351 encrypted connection does not ensure that the requested action will succeed. See
- 1352 OCF Security – Access Control section for more information.

1353 Example 1: below shows the Policy Parameter for a Resource that is discoverable but not

1354 observable, and for which authenticated accesses shall be done via CoAPS port 33275::

```

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

### 1357 7.8.2.1.3 “type” or Media Type Parameter

1358 The “type” Parameter may be used to specify the various media types that are supported by a

1359 specific target Resource. The default type of "application/cbor" shall be used when the “type”

1360 element is omitted. Once a Client discovers this information for each Resource, it may use one of

1361 the available representations in the appropriate header field of the Request or Response.

### 1362 7.8.2.1.4 “bp” or the Batch Interface Parameter

1363 The “batch” Parameter "bp" is used to specify the modifications to the target URI as the "batch"

1364 Request is forwarded through this Link. The "q" element in the value defines the query string that

1365 shall be appended to the "href" to make the target URI. The "q" query string may contain Property

1366 strings that are valid in that context. For example: Given a Collection as follows

```

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

1367 The following is the sequence for batch request to /room2

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

1368 See the Interfaces section 7.5 for more details on the "batch" Interface.

#### 1369 **7.8.2.1.5 "di" or Device ID parameter**

1370 The "di" Parameter specifies the device ID of the Device that hosts the target Resource defined in  
1371 the in the "href" Parameter.

1372 The device ID may be used to qualify a relative reference used in the "href" or to lookup endpoint  
1373 information for the relative reference.

#### 1374 **7.8.2.1.6 "buri" or base URI Parameter**

1375 The "buri" Parameter is the base URI to which the relative reference in "href" is resolved to. The  
1376 base URI and relative reference may be used to construct the URI to the target for the Link. The  
1377 base URI shall use the OCF Scheme for the URI defined in section 6.

#### 1378 **7.8.2.2 Formatting**

1379 When formatting in JSON, the list of Links shall be an array. The first element of the array shall be  
1380 a JSON object called the "tags block". This object may be empty or have keys that are the  
1381 Parameters from the list of Parameters for the Link. The "href" parameter shall not appear in the  
1382 "tags block". The second element of this array shall be a list of Links.

1383 For each list of Links the Parameters that appear in the "tags block" shall apply to each of the links  
1384 in the list of Links array associated with this tags block.

1385 A null list of Links shall have a null "tags block" and both shall not be included.

1386 NOTE: By this organization the list of Links is recursive and the "tags block" allows for a compact representation where  
1387 Parameters shared by multiple Links don't need to be repeated in each Links and can be factored into the "tags block".

1388 For example a list of Links with "tags" block.

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

```

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

```

1389

**Figure 13: Example “list of Links”**

1390

### 7.8.2.3 List of Links in a Collection

1391

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

1392

1393

A Resource with a list of Links

```

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

```

1394

**Figure 14: List of Links in a Resource**

#### 1395 **7.8.2.4 Usage Cases – Resource discovery**

1396 The OCF architecture utilizes typed Links as a mechanism for bootstrapping Resource discovery  
1397 through the known Core Resource /oic/res. A RETRIEVE operation on /oic/res returns (among  
1398 other things) a serialized representation of typed Links to Resources that are discoverable on that  
1399 Device.

1400 The serialization format should be negotiated using the underlying transport protocol (i.e. using  
1401 Accept and Content-Type headers in case of CoAP). By default, OCF uses CBOR as the payload.  
1402 The payload (content) in CBOR for Links is described with the JSON Schema in D.2.8. Other  
1403 serializations (e.g. XML/EXI) may be defined in future versions of this specification. The JSON  
1404 Schema that specifies the representation of the response to /oic/res is defined D.8.

### 1405 **7.8.3 Collections**

#### 1406 **7.8.3.1 Overview**

1407 A Resource that contains one or more references (specified as Links) to other resources is an  
1408 Collection. These reference may be related to each other or just be a list; the Collection provides  
1409 a means to refer to this set of references with a single handle (i.e. the URI). A simple resource is  
1410 kept distinct from a collection. Any Resource may be turned into an Collection by binding resource  
1411 references as Links. Collections may be used for creating, defining or specifying hierarchies,  
1412 indexes, groups, and so on.

1413 A Collection shall have at least one Resource Type and at least one Interface bound at all times  
1414 during its lifetime. During creation time of a collection the resource type and interfaces are  
1415 specified. The initial defined resource types and interfaces may be updated during its life time.  
1416 These initial values may be overridden using mechanism used for overriding in the case of a  
1417 Resource. Additional resource types and interfaces may be bound to the Collection at creation or  
1418 later during the lifecycle of the Collection.

1419 A Collection shall define the “links” Common Property. The value of the “links” Property is an array  
1420 with zero or more Links. The target URIs in the Links may reference another Collection or another  
1421 Resource. The referenced Collection or Resource may reside on the same Device as the Collection  
1422 that includes that Link (called a local reference) or may reside on another Device (called a remote  
1423 reference). The context URI of the Links in the “links” array shall (implicitly) be the Collection that  
1424 contains that “links” property. The (implicit) context URI may be overridden with explicit  
1425 specification of the “anchor” parameter in the Link where the value of “anchor” is the new base of  
1426 the Link.

1427 A Resource may be referenced in more than one Collection, therefore, a unique parent-child  
1428 relationship is not guaranteed. There is no pre-defined relationship between a Collection and the  
1429 Resource referenced in the Collection, i.e., the application may use Collections to represent a  
1430 relationship but none is automatically implied or defined. The lifecycles of the Collection and the  
1431 referenced Resource are also independent of one another.

1432 If the “drel” property is defined for the Collection then all Links that don’t explicitly specify a  
1433 relationship shall inherit this default relationship in the context of that Collection. The default  
1434 relationship defines the implicit relationship between the Collection and the target URI in the Link.

1435 The list of Links defined in a Collection may be either a simple list of Links as illustrated in Figure  
1436 16 or may be a list of tagged Links sets as illustrated in Figure 17. For the former, the value of the  
1437 “links” Property is a simple array of Links. For the later, the value of the “links” Property is an array  
1438 where each element is a resource containing a Links array and a set of one or more key-value  
1439 pairs; the key-value pairs are the tags for the Links array (the key is the tag name and the value  
1440 is the tag value)



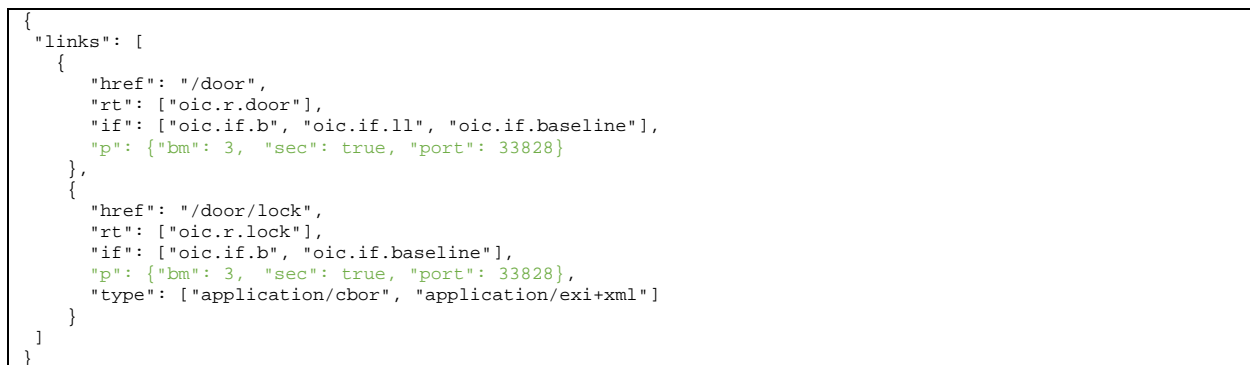


1441

1442

**Figure 15: Example showing parts of Collection and Links**

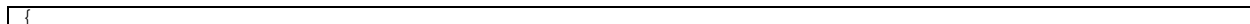
1443



1444

**Figure 16: Example Collection with simple links (JSON)**

1445



```

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

```

**Figure 17: Example Collection with tagged Links (JSON)**

1446

1447 Note: Example shows only one tag; each tag has the same tag name, i.e., "di", but have different tag values.

1448

1449 A Collection may be:

- 1450 • A pre-defined Collection where the Collection has been defined a priori and the Collection is
- 1451 static over its lifetime. Such Collections may be used to model, for example, an appliance that
- 1452 is composed of other devices or fixed set of resource representing fixed functions.
- 1453 • A Device local Collection where the Collection is used only on the Device that hosts the
- 1454 Collection. Such collections may be used as a short-hand on a client for referring to many
- 1455 Servers as one.
- 1456 • A centralized Collection where the Collection is hosted on an Device but other Devices may
- 1457 access or update the Collection
- 1458 • A hosted Collection where the collection is centralized but is managed by an authorized agent
- 1459 or party.

1460 **7.8.3.2 Collection Properties**

1461 An Collection shall define the "links" Property. In addition, other Properties may be defined for the

1462 Collection by the Resource Type. The mandatory and recommended Common Properties for

1463 Collection are shown in Table 9. This list of Common Properties are in addition to those defined

1464 for Resources in section 7.3.2. When a property is repeated in Table 9 , the conditions in this

1465 definition shall override those in the general list for Resources.

1466  
1467

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

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

1468

1469 The Properties of a Collection may not be modified.

1470 **7.8.3.3 Default resource type**

1471 A default Resource Type, oic.wk.col, shall be available for Collections. This Resource Type shall  
1472 be used only when another type has not been defined on the Collection or when no Resource Type  
1473 has been specified at the creation of the Collection.

1474 The default Resource Type provides support for the Common Properties including the "links"  
1475 Property. For the default resource type, the value of "links" shall be a simple array of Links and  
1476 tagging of links shall not be supported.

1477 The default Resource Type shall support the 'baseline' and 'links list' Interfaces. The default  
1478 Interface shall be the 'links list' Interface.

1479 **8 CRUDN**

1480 **8.1 Overview**

1481 CREATE, RETRIEVE, UPDATE, DELETE, and NOTIFY (CRUDN) are operations defined for  
1482 manipulating Resources. These operations are performed by a Client on the resources contained  
1483 in an Server.

1484 On reception of a valid CRUDN operation an Server hosting the Resource that is the target of the  
1485 request shall generate a response depending on the Interface included in the request; or based  
1486 on the Default Interface for the Resource Type if no Interface is included.

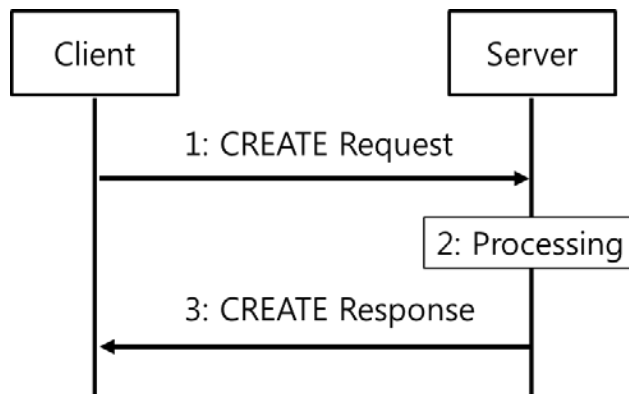
1487 CRUDN operations utilize a set of parameters that are carried in the messages and are defined in  
 1488 Table 10. A Device shall use CBOR as the default payload (content) encoding scheme for resource  
 1489 representations included in CRUDN operations and operation responses; a Device may negotiate  
 1490 a different payload encoding scheme (e.g, see in section 12.2.4 for CoAP messaging). The  
 1491 following subsections specify the CRUDN operations and use of the parameters. The type  
 1492 definitions for these terms will be mapped in the messaging section for each protocol.

1493 **Table 10. Parameters of CRUDN messages**

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

1494 **8.2 CREATE**

1495 The CREATE operation is used to request the creation of new Resources on the Server. The  
 1496 CREATE operation is initiated by the Client and consists of three steps, as depicted in Figure 18  
 1497 and described below.



1498

1499

Figure 18. CREATE operation

1500 **8.2.1 CREATE request**

1501 The CREATE request message is transmitted by the Client to the Server to create a new Resource  
1502 by the Server. The CREATE request message will carry the following parameters:

- 1503 • *fr*: Unique identifier of the Client
- 1504 • *to*: URI of the target resource responsible for creation of the new resource.
- 1505 • *ri*: Identifier of the CREATE request
- 1506 • *cn*: Information of the resource to be created by the Server
  - 1507 i) *cn* will include the URI and resource type property of the resource to be created.
  - 1508 ii) *cn* may include additional properties of the resource to be created.
- 1509 • *op*: CREATE

1510 **8.2.2 Processing by the Server**

1511 Following the receipt of a CREATE request, the Server may validate if the Client has the  
1512 appropriate rights for creating the requested resource. If the validation is successful, the Server  
1513 creates the requested resource. The Server caches the value of *ri* parameter in the CREATE  
1514 request for inclusion in the CREATE response message.

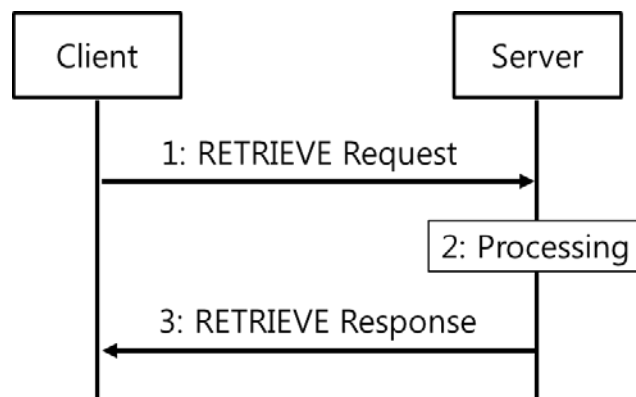
1515 **8.2.3 CREATE response**

1516 The Server shall transmit a CREATE response message in response to a CREATE request  
1517 message from a Client. The CREATE response message will include the following parameters.

- 1518 • *fr*: Unique identifier of the Server
- 1519 • *to*: Unique identifier of the Client
- 1520 • *ri*: Identifier included in the CREATE request
- 1521 • *cn*: Information of the resource as created by the Server.
  - 1522 i) *cn* will include the URI of the created resource.
  - 1523 ii) *cn* will include the resource representation of the created resource.
- 1524 • *rs*: The result of the CREATE operation

1525 **8.3 RETRIEVE**

1526 The RETRIEVE operation is used to request the current state or representation of a Resource.  
1527 The RETRIEVE operation is initiated by the Client and consists of three steps, as depicted in  
1528 Figure 19 and described below.



1529

1530

**Figure 19. RETRIEVE operation**

1531 **8.3.1 RETRIEVE request**

1532 RETRIEVE request message is transmitted by the Client to the Server to request the  
1533 representation of a Resource from an Server. The RETRIEVE request message will carry the  
1534 following parameters.

- 1535 • *fr*: Unique identifier of the Client
- 1536 • *to*: URI of the resource the Client is targeting
- 1537 • *ri*: Identifier of the RETRIEVE request
- 1538 • *op*: RETRIEVE

1539 **8.3.2 Processing by the Server**

1540 Following the receipt of a RETRIEVE request, the Server may validate if the Client has the  
1541 appropriate rights for retrieving the requested data and the properties are readable. The Server  
1542 caches the value of *ri* parameter in the RETRIEVE request for use in the response.

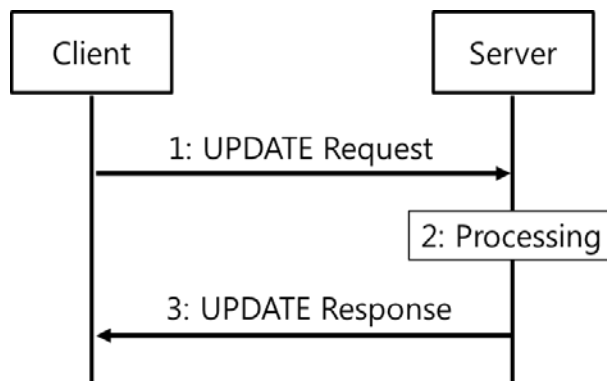
1543 **8.3.3 RETRIEVE response**

1544 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request  
1545 message from a Client. The RETRIEVE response message will include the following parameters.

- 1546 • *fr*: Unique identifier of the Server
- 1547 • *to*: Unique identifier of the Client
- 1548 • *ri*: Identifier included in the RETRIEVE request
- 1549 • *cn*: Information of the resource as requested by the Client
  - 1550 i) *cn* should include the URI of the resource targeted in the RETRIEVE request
- 1551
- 1552 • *rs*: The result of the RETRIEVE operation

1553 **8.4 UPDATE**

1554 The UPDATE operation is either a Partial UPDATE or a complete replacement of the information  
1555 in a Resource in conjunction with the interface that is also applied to the operation. The UPDATE  
1556 operation is initiated by the Client and consists of three steps, as depicted in Figure 20 and  
1557 described below.



1558

**Figure 20. UPDATE operation**

1559

1560 **8.4.1 UPDATE request**

1561 The UPDATE request message is transmitted by the Client to the Server to request the update of  
1562 information of a Resource on the Server. The UPDATE request message will carry the following  
1563 parameters.

- 1564 • *fr*: Unique identifier of the Client
- 1565 • *to*: URI of the resource targeted for the information update
- 1566 • *ri*: Identifier of the UPDATE request
- 1567 • *op*: UPDATE
- 1568 • *cn*: Information, including properties, of the resource to be updated at the target resource

1569 **8.4.2 Processing by the Server**

1570 Following the receipt of an UPDATE request, the Server may validate if the Client has the  
1571 appropriate rights for updating the requested data. If the validation is successful the Server  
1572 updates the target Resource information according to the information carried in *cn* parameter of  
1573 the UPDATE request message. The Server caches the value of *ri* parameter in the UPDATE  
1574 request for use in the response.

1575 An UPDATE request that includes Properties that are read-only shall be rejected by the Server  
1576 with an *rs* indicating a bad request.

1577 An UPDATE request shall be applied only to the Properties in the target resource visible via the  
1578 applied interface that support the operation. An UPDATE of non-existent Properties is ignored.

1579 **8.4.3 UPDATE response**

1580 The UPDATE response message will include the following parameters:

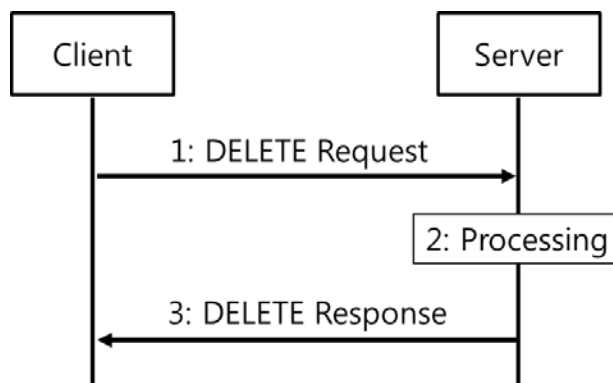
- 1581 • *fr*: Unique identifier of the Server
- 1582 • *to*: Unique identifier of the Client
- 1583 • *ri*: Identifier included in the UPDATE request
- 1584 • *rs*: The result of the UPDATE request

1585 The UPDATE response message may also include the following parameters:

- 1586 • *cn*: The Resource representation following processing of the UPDATE request

1587 **8.5 DELETE**

1588 The DELETE operation is used to request the removal of a Resource. The DELETE operation is  
1589 initiated by the Client and consists of three steps, as depicted in Figure 21 and described below.



1590 **Figure 21. DELETE operation**

### 1592 **8.5.1 DELETE request**

1593 DELETE request message is transmitted by the Client to the Server to delete a Resource on the  
1594 Server. The DELETE request message will carry the following parameters:

- 1595 • *fr*: Unique identifier of the Client
- 1596 • *to*: URI of the target resource which is the target of deletion
- 1597 • *ri*: Identifier of the DELETE request
- 1598 • *op*: DELETE

### 1599 **8.5.2 Processing by the Server**

1600 Following the receipt of a DELETE request, the Server may validate if the Client has the  
1601 appropriate rights for deleting the identified resource, and whether the identified resource exists.  
1602 If the validation is successful, the Server removes the requested resource and deletes all the  
1603 associated information. The Server caches the value of *ri* parameter in the DELETE request for  
1604 use in the response.

### 1605 **8.5.3 DELETE response**

1606 The Server shall transmit a DELETE response message in response to a DELETE request  
1607 message from a Client. The DELETE response message will include the following parameters.

- 1608 • *fr*: Unique identifier of the Server
- 1609 • *to*: Unique identifier of the Client
- 1610 • *ri*: Identifier included in the DELETE request
- 1611 • *rs*: The result of the DELETE operation

## 1612 **8.6 NOTIFY**

1613 The NOTIFY operation is used to request asynchronous notification of state changes. Complete  
1614 description of the NOTIFY operation is provided in section 11.4. The NOTIFY operation uses the  
1615 NOTIFICATION response message which is defined here.

### 1616 **8.6.1 NOTIFICATION response**

1617 The NOTIFICATION response message is sent by an Server to notify the URLs identified by the  
1618 Client of a state change. The NOTIFICATION response message carries the following parameters.

- 1619 • *fr*: Unique identifier of the Server
- 1620 • *to*: URI of the Resource target of the NOTIFICATION message
- 1621 • *ri*: Identifier included in the CREATE request
- 1622 • *op*: NOTIFY
- 1623 • *cn*: The updated state of the resource

## 1624 **9 Network and connectivity**

### 1625 **9.1 Introduction**

1626 The IOT environment, which the OCF is addressing, is composed of very heterogeneous systems.  
1627 Because these systems are often tailored to address dedicated requirements, they are composed  
1628 of very diverse products and services. Those products span from very constrained devices that  
1629 run on batteries to every day high end devices available on consumer market shelves. The lack of  
1630 a global standard and the need to create such a standard has led various groups to work on  
1631 streamlining those technologies with well-established networking standards.

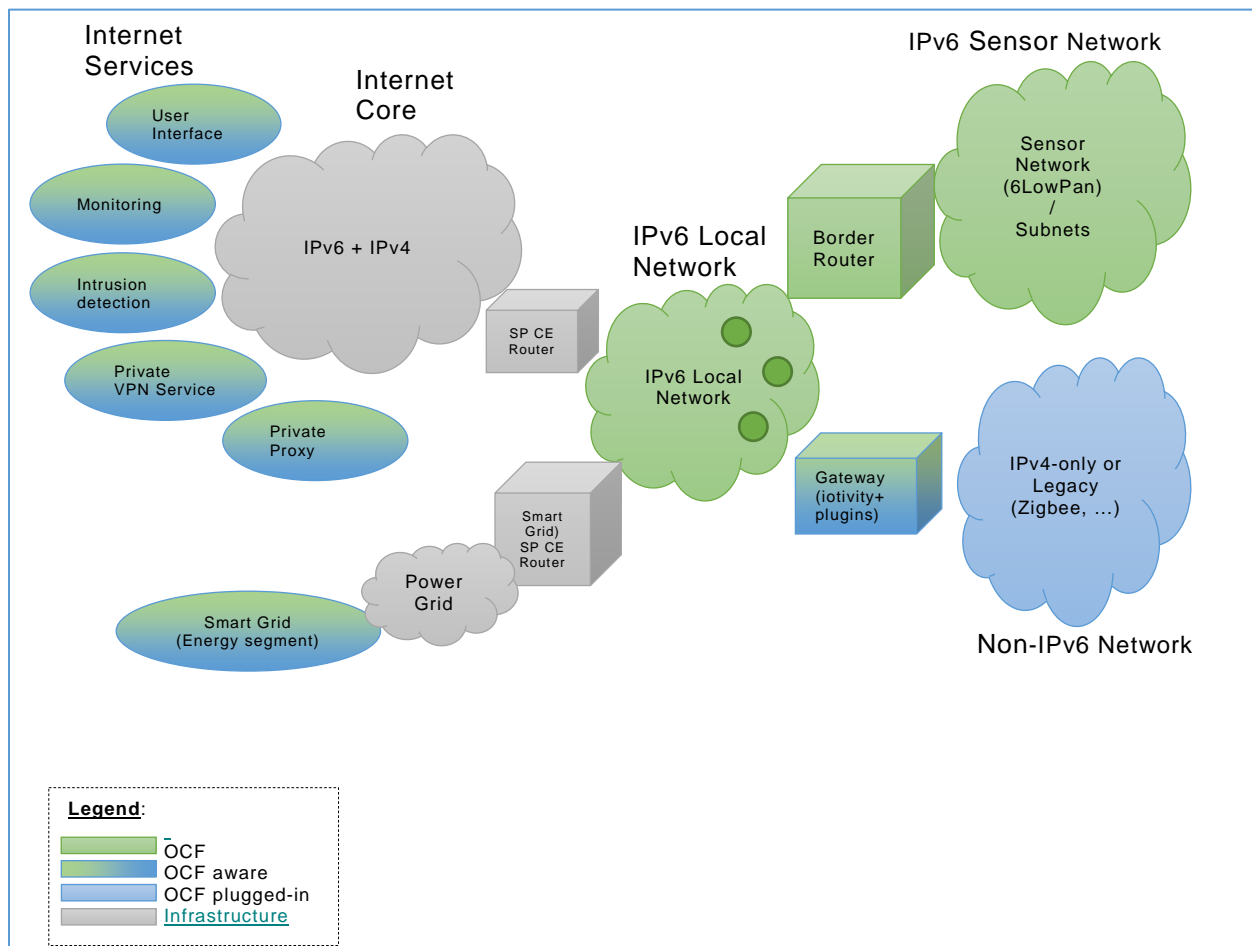


1632 The IETF recognized the market transition and realized that Ipv4 was no longer adequate. Not only  
 1633 does the new scale call for a new technology, but also the manageability of even more diverse  
 1634 devices, the complexity of multiple subnets and higher security and privacy needs require a whole  
 1635 new set of standards. Cognizant of the existence and need for dedicated physical layer and data  
 1636 link layer, the IETF set up working groups to streamline the various existing technologies at the  
 1637 network layer. In accordance with these market realities, this specification also means to leverage  
 1638 existing radio silicon (e.g., Bluetooth® technology, Wi-Fi, or 802.15.4) and concentrates on the  
 1639 network layer and the associated data link layer adaptations produced by the IETF.

1640 **9.2 Architecture**

1641 While the aging IPv4 centric network has evolved to support complex topologies, its deployment  
 1642 was primarily provisioned by a single Internet Service Provider (ISP) as a single network. More  
 1643 complex network topologies, often seen in residential home, are mostly introduced through the  
 1644 acquisition of additional home network devices, which rely on technologies like private Network  
 1645 Address Translation (NAT). These technologies require expert assistance to set up correctly and  
 1646 should be avoided in a home network as they most often result in breakage of constructs like  
 1647 routing, naming and discovery services.

1648 The multi-segment ecosystem OCF addresses will not only cause a proliferation of new devices  
 1649 and associated routers, but also new services introducing additional edge routers. All these new  
 1650 requirements require advance architectural constructs to address complex network topologies like  
 1651 the one shown in Figure 22.



1652

1653

**Figure 22. High Level Network & Connectivity Architecture**

1654 In terms of RFC 6434, IPv6 nodes assume either a router or host role. Nodes may further  
1655 implement various specializations of those roles. In terms of RFC 6434, IPv6 nodes assume either a  
1656 router or host role. Nodes may further implement various specializations of those roles:  
1657  
1658 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.  
1659  
1660 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity  
1661 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).  
1662 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU  
1663 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),  
1664  
1665 • A node may translate and route messaging between IPv6 and non-IPv6 networks.  
1666  
1667 • A Router may implement Customer Edge Router capabilities as defined in IETF RFC 7084.  
1668  
1669 • Nodes limited in processing power, memory, non-volatile storage or transmission capacity  
1670 requires special IP adaptation layers (6LoWPAN) and/or dedicated routing protocols (RPL).  
1671 Examples include devices transmitting over low power physical layer like IEEE 802.14.5, ITU  
1672 G9959, Bluetooth Low Energy, DECT Ultra Low Energy, Near Field Communication (NFC),  
1673

### 1674 **9.3 • A node may translate and route messaging between IPv6 and non-IPv6** 1675 **networks. IPv6 network layer requirements**

#### 1676 **9.3.1 Introduction**

1677 Projections indicate that many 10s of billions of new IoT endpoints and related services will be  
1678 brought online in the next few years. These endpoint's capabilities will span from battery powered  
1679 nodes with limited compute, storage, and bandwidth to more richly resourced devices operating  
1680 over Ethernet and WiFi links.

1681 Internet Protocol version 4 (IPv4), deployed some 30 years ago, has matured to support a wide  
1682 variety of applications such as Web browsing, email, voice, video, and critical system monitoring  
1683 and control. However, the capabilities of IPv4 are at the point of exhaustion, not the least of which  
1684 is that available address space has been consumed.

1685 The IETF long ago saw the need for a successor to IPv4, thus the development of IPv6. OCF  
1686 recommends IPv6 at the network layer. Amongst the reasons for IPv6 recommendations are:

- 1687 • Larger address space. Side-effect: greatly reduce the need for NATs.
- 1688 • More flexible addressing architecture. Multiple addresses and types per interface: Link-local,  
1689 ULA, GUA, variously scoped Multicast addresses, etc. Better ability to support multi-homed  
1690 networks, better re-numbering capability, etc.
- 1691 • More capable auto configuration capabilities: DHCPv6, SLAAC, Router Discovery, etc.
- 1692 • Technologies enabling IP connectivity on constrained nodes are based upon IPv6.
- 1693 • All major consumer operating systems (iOS, Android, Windows, Linux) are already IPv6 enabled.
- 1694 • Major Service Providers around the globe are deploying IPv6.

#### 1695 **9.3.2 IPv6 node requirements**

##### 1696 **9.3.2.1 Introduction**

1697 In order to ensure network layer services interoperability from node to node, mandating a common  
1698 network layer across all nodes is vital. The protocol should enable the network to be: secure,  
1699 manageable, scalable and to include constrained and self-organizing meshed nodes. OCF  
1700 recommends IPv6 as the common network layer protocol to ensure interoperability across all  
1701 Devices. More capable devices may also include additional protocols creating multiple-stack

1702 devices. The remainder of this section will focus on interoperability requirements for IPv6 hosts,  
1703 IPv6 constrained hosts and IPv6 routers. The various protocol translation permutations included  
1704 in multi-stack gateway devices may be addresses in subsequent addendums of this specification.

### 1705 **9.3.2.2 IP Layer**

1706 An IPv6 node should support IPv6. If a node supports IPv6, then it shall conform to the  
1707 requirements for communication on the local network as follows:

- 1708 • Shall support IETF RFC 2460 “Internet Protocol version 6 Specification” and related updates  
1709 as defined in section 5.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1710 • Shall support IETF RFC 4291 “IP Version 6 Addressing Architecture” and related updates as  
1711 defined in section 5.9.1 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1712 • Shall support IETF RFC 4861 “Neighbor Discovery for IPv6” and related updates as defined in  
1713 section 5.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1714 • Shall support IETF RFC 4862 “IPv6 Stateless Address Autoconfiguration” and related updates  
1715 as defined in section 5.9.2 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1716 • Shall support IETF RFC 4443 “Internet Control Message Protocol (ICMPv6) for IPv6” [RFC4443]  
1717 and related updates as defined in section 5.8 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1718 • Shall support IETF RFC 1981 “Path MTU Discovery” and related updates as defined in section  
1719 5.6 of IETF RFC 6434 “IPv6 Node Requirements”.
- 1720 • Shall support IETF RFC 4193 “Unique Local IPv6 Unicast Addresses” and related updates.
- 1721 • Shall support IETF RFC 3810 “Multicast Listener Discovery Version 2 (MLDv2) for IPv6” and  
1722 related updates. In particular, shall generate new MLDv2 Report messages for every “All OCF  
1723 Nodes” address FF0X::158 joined on an interface.

1724 .

## 1725 **9.3.3 IPv6 constrained nodes**

### 1726 **9.3.3.1 Requirements**

1727 An IPv6 constrained node shall support all node requirements defined in section 9.3.2. If a  
1728 constrained node supports IPv6, it should use the adaptations defined as follows in order to support  
1729 IPv6.

### 1730 **9.3.3.2 IP layer**

1731 An IPv6 constrained node should support the neighbour discovery optimization as defined in  
1732 IETF RFC 6775 “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal  
1733 Area Networks (6LoWPANs)”.

### 1734 **9.3.3.3 Sub IP layer**

- 1735 • An IPv6 constrained node on an ITU-T G.9959 network should support IETF RFC 7428 and  
1736 related updates.
- 1737 • An IPv6 constrained node on an IEEE 802.15.4 network should support IETF RFC 4944 and  
1738 related updates.
- 1739 • An IPv6 constrained node on a BLUETOOTH(R) Low Energy network should support  
1740 IETF RFC 7668 and related updates.

## 1741 10 Endpoint discovery

### 1742 10.1 Introduction

1743 This section describes how an OCF Endpoint is discovered by another OCF Endpoint in a network.  
1744 An OCF Endpoint shall support CoAP discovery.

### 1745 10.2 CoAP based Endpoint discovery

1746 The following describes CoAP based Endpoint discovery:

- 1747 a) Advertising or publishing Devices shall join the 'All OCF Nodes' multicast groups (as defined  
1748 in [IANA IPv6 Multicast Address Space Registry]) and listen on the port 5683.
- 1749 b) Clients intending to discover resources shall join the 'All OCF Nodes' multicast groups (as  
1750 defined in [IANA IPv6 Multicast Address Space Registry]).
- 1751 c) Clients shall send discovery requests (GET request) to the 'All OCF Nodes' multicast group  
1752 address at port 5683. The requested URI shall be /oic/res.
- 1753 d) If the discovery request is intended for a specific resource type, the Query parameter "rt" shall  
1754 be included in the request (section 6.2.1) with its value set to the desired resource type. Only  
1755 Devices hosting the resource type shall respond to the discovery request.
- 1756 e) When the "rt" Query parameter is omitted, all Devices shall respond to the discovery request.
- 1757 f) Handling of multicast requests shall be as described in section 8 of IETF RFC 7252 and section  
1758 4.1 in IETF RFC 6690.
- 1759 g) Devices which receive the request shall respond using CBOR payload encoding. A Device  
1760 shall indicate support for CBOR payload encoding for multicast discovery as described in  
1761 section 12.2.3. Later versions of the specification may support alternate payload encodings  
1762 (JSON, XML/EXI, etc.).

1763

1764 Below are a few examples to search for Devices on the network:

1765 To search for all Devices on the network a Client can issue:

#### 1766 Request

1767 GET /oic/res

#### 1768 Response

```
1769 [
1770 {
1771   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1772   "links": [
1773     {
1774       "href": "/oic/d",
1775       "rt": ["oic.d.light", "oic.wd.d"],
1776       "if": ["oic.if.r", "oic.if.baseline"],
1777       "p": {"bm": 1, "sec": true, "port": 32278}
1778     },
1779     {
1780       "href": "/oic/p",
1781       "rt": ["oic.wk.p"],
1782       "if": ["oic.if.r", "oic.if.baseline"],
1783       "p": {"bm": 1, "sec": true, "port": 32278}
1784     },
1785     {
1786       "href": "/switch",
1787       "rt": ["oic.r.switch.binary"],
1788       "if": ["oic.if.a", "oic.if.baseline"],
1789       "p": {"bm": 2, "sec": true, "port": 32278}
1790     },
1791     {
1792       "href": "/brightness",
```

```

1793     "rt": ["oic.r.light.brightness"],
1794     "if": ["oic.if.a", "oic.if.baseline"],
1795     "p": {"bm": 3, "sec": true, "port": 32278}
1796   }
1797 ]
1798 }
1799 ]

```

1800 To search for oic.r.switch.binary resources on the network a Client can issue:

1801 **Request**

1802 GET /oic/res?rt=oic.r.switch.binary

1803 **Response**

```

1804 [
1805 {
1806   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1807   "links": [
1808     {
1809       "href": "/switch",
1810       "rt": ["oic.r.switch.binary"],
1811       "if": ["oic.if.a", "oic.if.baseline"],
1812       "p": {"bm": 1, "sec": true, "port": 32278}
1813     }
1814   ]
1815 }
1816 ]

```

1817 Note that the examples do not indicate the multicast address and port number. The examples also do not include the  
1818 accept header.

1819

1820 **11 Functional interactions**

1821 **11.1 Introduction**

1822 The functional interactions between a Client and n Server are described in section 11.2 through  
1823 section 11.6 respectively. The functional interactions use CRUDN messages (section 8) and  
1824 include Discovery, Notification, and Device management. These functions require support of core  
1825 defined resources as defined in Table 11. More details about these resources are provided later  
1826 in this section.

1827

**Table 11. List of Core Resources**

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

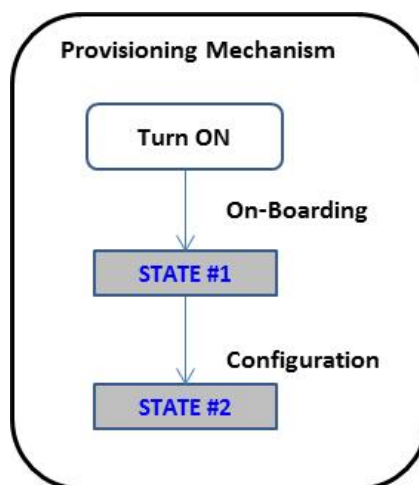
1828

1829 **11.2 Provisioning**

1830 Provisioning in Framework includes two distinct processes: onboarding and Configuration.

1831 onboarding is the process which delivers required information to a Device for joining the OCF  
1832 network. When onboarding process is completed, the Device has necessary information and is  
1833 able to join the OCF network (State #1 in Figure 23). Further details about provisioning can be  
1834 found in OCF Security specification (Owner PSK).

1835 Configuration is the process which delivers required information to a device for accessing OCF  
1836 services. At the end of the configuration process, the Device has all the necessary information and  
1837 is able to access OCF services (State #2 in Figure 23).



1838

1839

**Figure 23. Provisioning State Changes**

#### 1840 #1 onboarding

1841 Framework is applicable to many different types of devices with different capabilities, including  
1842 devices with a rich user interface that can take inputs from the users, e.g., smartphones, as well  
1843 as headless devices that have no means for receiving user inputs, e.g., sensors. Additionally, the  
1844 Devices may support different communication and connectivity technologies, e.g., Bluetooth, Wi-  
1845 Fi, etc. Different communication and connectivity technologies provide different onboarding  
1846 mechanisms specific to that technology.

1847 Due to these differences and diversity of device capabilities, this version of specification does not  
1848 mandate a particular process for onboarding, instead, specifies the state of the Device upon  
1849 completion of the onboarding process.

1850 As part of the onboarding process the device acquires detailed information and required parameter  
1851 values to be able to connect to the network, resulting in successful establishment of a connection  
1852 to the network at the end of the onboarding process. The required information and parameters  
1853 values include for example, SSID for Wi-Fi as well as authentication credentials.

1854 Later versions of this specification may specify a common process for onboarding across different  
1855 communication and connectivity technologies.

#### 1856 #2 Configuration

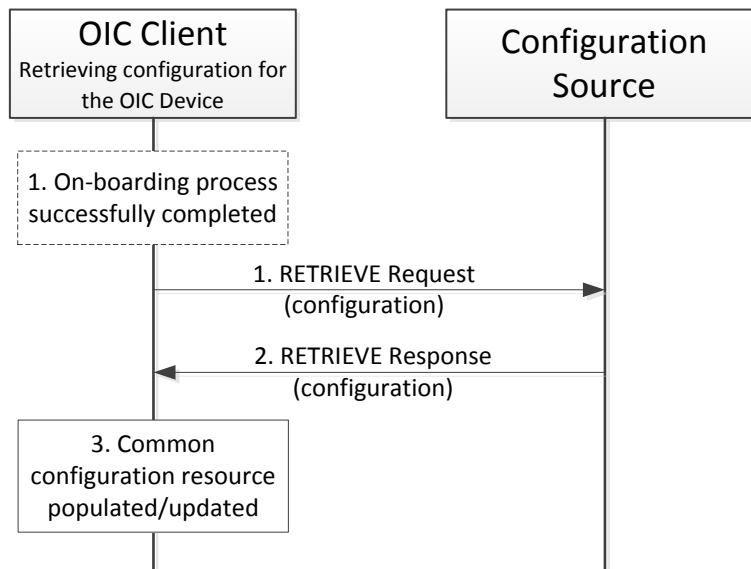
1857 Once a Device is successfully connected to the OCF network, it needs additional configuration  
1858 information for accessing the OCF services or to subscribe for OCF services. The information  
1859 required may include geographical location, time zone, security requirements, etc. This information  
1860 may be pre-loaded on an Device, or may be acquired from a configuration service that can be  
1861 located on another Device, e.g., the Configuration Source. The information regarding the  
1862 configuration service resource, e.g., the URI of the Configuration Source, is pre-configured on the  
1863 Device.

1864 The configuration information is also in core resource /oic/con. Upon completion of the onboarding  
1865 process and as soon as the Device is connected to the network, if the configuration information is  
1866 not pre-loaded, it shall initiate the configuration process, as a result of which the Device acquires  
1867 the relevant configuration information, through either a pull or a push interaction, and populates  
1868 its designated configuration resource with its current configured state information. The designated  
1869 configuration resource maintains the latest configuration state and is the designated resource  
1870 through which updates to the configuration are made.

1871 If the configuration information is not pre-loaded the Device retrieves them from the Configuration  
1872 Source. During the lifetime of a Device a Client may retrieve or update the configuration state of  
1873 the Device. Some of the configuration information is read only and some may be modified by  
1874 Configuration Sources depending on the 'Access Modes' of properties in /oic/con resource.

1875 Figure 24 depicts the interactions triggered by a Device to retrieve its configuration information  
1876 from the Configuration Source which may be located on a remote Device or locally. These  
1877 interactions occur instantly following completion of onboarding process; the Device may retrieve  
1878 its configuration at any time during its lifetime.

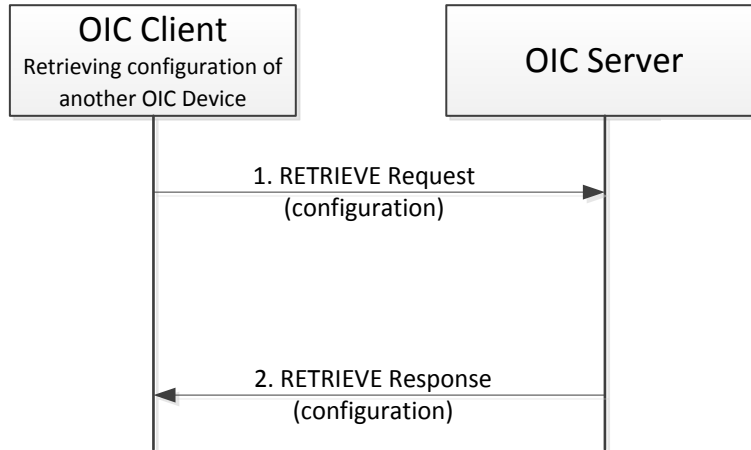
1879



1880

1881 **Figure 24. Interactions initiated by the Device to retrieve its configuration from a**  
1882 **configuration source**

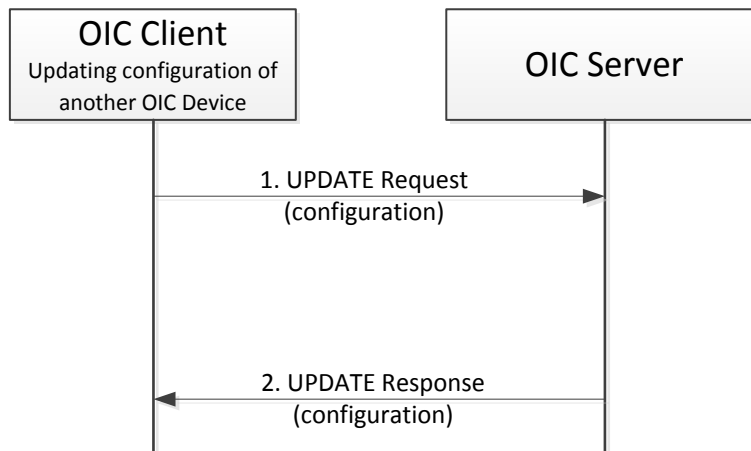
1883 Figure 25 depicts the interactions when the retrieve of configuration information is done by a Client.



1884  
1885

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

1886 Figure 26 depicts the interactions when the configuration information of an Device is updated by a  
1887 Client, e.g., the Configuration Source.  
1888



1889  
1890

**Figure 26. Update of and Device configuration**

1891 If Configuration is supported by a Device, i.e., the configuration information may be dynamically  
1892 updated, the Core Resource /oic/con shall be supported as the designated configuration resource  
1893 as described in Table 12.

1894 **Configuration Resource**

1895 A Device or a Platform may be initially configured from information that is set or provisioned at  
1896 bootstrap. In addition, the Device and Platform may be configured further by an external agent  
1897 post bootstrap depending on changing conditions or context. The core resource /oic/con exposes  
1898 properties that may be used to effect changes in the configuration.  
1899

1900 A configuration is determined by setting all the properties that collectively pertain to that  
1901 configuration. The outcome of setting a new configuration is determined by the value of the specific  
1902 properties in that set. Setting a new configuration through /oic/con may lead to initiation of  
1903 processes that affect or create side effects in other resources.

1904



1905

**Table 12. Configuration Resources**

| Pre-defined URI | Resource Type Title | Resource Type ID ("rt" value) | Interfaces | Description   | Related Functional Interaction |
|-----------------|---------------------|-------------------------------|------------|---|--------------------------------|
| /oic/con        | Configuration       | oic.wk.con                    | oic.if.rw  | The resource through which configurable information specific to the Device is exposed. The <b>resource properties</b> exposed by /oic/con are listed in Table 13. | Configuration                  |

1906

1907 Table 13 defines the oic.wk.con resource type.

1908

1909

**Table 13. oic.wk.con resource type definition**

| Property title       | Property name | Value type   | Value rule | Unit | Access mode | Mandatory | Description  |
|----------------------|---------------|--|------------|------|-------------|-----------|--|
| <b>(Device) Name</b> | n             | string   |            |      | R, W        | yes       | Human friendly name configurable by the end user (e.g. Bob's thermostat).  |
| <b>Location</b>      | loc           | json (has two attributes one with longitude and latitude and also a name for a location) |            |      | R, W        | no        | Provides location information where available.   |
| <b>Location Name</b> | locn          | string   |            |      | R, W        | no        | Human friendly name for location For example, "Living Room".   |
| <b>Currency</b>      | c             | string   |            |      | R,W         | no        | Indicates the currency that is used for any monetary transactions  |
| <b>Region</b>        | r             | string   |            |      | R,W         | no        | Free form text Indicating the current region in which the device is located geographically. The free form text shall not start with a quote ("). |

1910

1911 **11.3 Resource discovery**

1912 **11.3.1 Introduction**

1913 Discovery is a function which enables endpoint discovery as well as resource based discovery.  
 1914 Endpoint discovery is described in detail in section 10. This section mainly describes the resource  
 1915 based discovery.

1916 **11.3.2 Resource based discovery: mechanisms**

1917 **11.3.2.1 Overview**

1918 As part of discovery, a Client may find appropriate information about other OCF peers. This  
 1919 information could be instances of resources, resource types or any other information represented  
 1920 in the resource model that an OCF peer would want another OCF peer to discover.

1921 At the minimum, Resource based discovery uses the following:

- 1922 1) A resource to enable discovery shall be defined. The representation of that resource shall  
1923 contain the information that can be discovered.
- 1924 2) The resource to enable discovery shall be specified and commonly known a-priori. A Device for  
1925 hosting the resource to enable discovery shall be identified.
- 1926 3) A mechanism and process to publish the information that needs to be discovered with the  
1927 resource to enable discovery.
- 1928 4) A mechanism and process to access and obtain the information from the resource to enable  
1929 discovery. A query may be used in the request to limit the returned information.
- 1930 5) A scope for the publication
- 1931 6) A scope for the access.
- 1932 7) A policy for visibility of the information.

1933

1934 Depending on the choice of the base aspects defined above, the Framework defines three resource  
1935 based discovery mechanisms:

- 1936 • Direct discovery, where the Resources are published locally at the Device hosting the  
1937 resources and are discovered through peer inquiry.
- 1938 • Indirect discovery, where Resources are published at a third party assisting with the  
1939 discovery and peers publish and perform discovery against the resource to enable  
1940 discovery on the assisting 3<sup>rd</sup> party.
- 1941 • Advertisement discovery, where the resource to enable discovery is hosted local to the  
1942 initiator of the discovery inquiry but remote to the Devices that are publishing discovery  
1943 information.

1944 A Device shall support direct discovery.

#### 1945 **11.3.2.2 Direct discovery**

1946 In direct discovery,

- 1947 1) The Device that is providing the information shall host the resource to enable discovery.
- 1948 2) The Device publishes the information available for discovery with the local resource to  
1949 enable discovery (i.e. local scope).
- 1950 3) Clients interested in discovering information about this Device shall issue RETRIEVE  
1951 requests directly to the resource. The request may be made as a unicast or multicast.  
1952 The request may be generic or may be qualified or limited by using appropriate queries in  
1953 the request.
- 1954 4) The “server” Device that receives the request shall send a response with the discovered  
1955 information directly back to the requesting “client” Device.
- 1956 5) The information that is included in the request is determined by the policies set for the  
1957 resource to be discovered locally on the responding Device.

1958

#### 1959 **11.3.2.3 Indirect discovery of Resources (resource directory based discovery)**

1960 In indirect discovery the information about the resource to be discovered is hosted on a Server  
1961 that is not hosting the resource. See section 11.3.6 for details on resource directory based  
1962 discovery.

1963 In indirect discovery:

- 1964 a) The resource to be discovered is hosted on a Device that is neither the client initiating  
1965 the discovery nor the Device that is providing or publishing the information to be  
1966 discovered. This Device may use the same resource to provide discovery for multiple  
1967 agents looking to discover and for multiple agents with information to be discovered.  
1968 b) The Device to be discovered or with information to discover, publishes that information  
1969 with resource to be discovered on a different Device. The policies on the information  
1970 shared including the lifetime/validity are specified by the publishing Device. The  
1971 publishing Device may modify these policies as required.  
1972 c) The client doing the discovery may send a unicast discovery request to the Device  
1973 hosting the discovery information or send a multicast request that shall be monitored and  
1974 responded to by the Device. In both cases, the Device hosting the discovery information  
1975 is acting on behalf of the publishing Device.  
1976 d) The discovery policies may be set by the Device hosting the discovery information or by  
1977 the party that is publishing the information to be discovered. The discovery information  
1978 that is returned in the discovery response shall adhere to the policies that are in effect at  
1979 the time of the request.  
1980

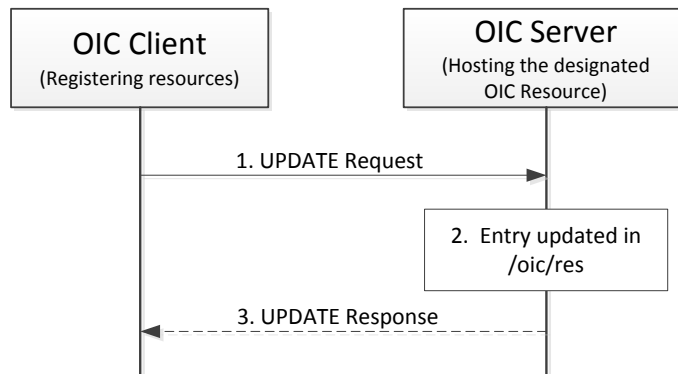
#### 1981 **11.3.2.4 Advertisement Discovery**

1982 In advertisement discovery:

- 1983 a) The resource to enable discovery is hosted local to the Device that is initiating the discovery  
1984 request (client). The resource to enable discovery may be an Core Resource or discovered  
1985 as part of a bootstrap.  
1986 b) The request could be an implementation dependent lookup or be a local RETRIEVE request  
1987 against the resource that enables discovery.  
1988 c) The Device with information to be discovered shall publish the appropriate information to  
1989 the resource that enables discovery.  
1990 d) The publishing Device is responsible for the published information. The publishing Device  
1991 may UPDATE the information at the resource to enable discovery based on its needs by  
1992 sending additional publication requests. The policies on the information that is discovered  
1993 including lifetime is determined by the publishing Device.  
1994

#### 1995 **11.3.3 Resource based discovery: Information publication process**

1996 The mechanism to publish information with the resource to enable discovery can be done either  
1997 locally or remotely. The publication process is depicted in Figure 27. The Device which has  
1998 discovery information to publish shall a) either update the resource that enables discovery if  
1999 hosted locally or b) issue an UPDATE request with the information to the Device which hosts the  
2000 resource that enables discovery. The Device hosting the resource to enable discovery  
2001 adds/updates the resource to enable discovery with the provided information and then responds  
2002 to the Device which has requested the publication of the resource with an UPDATE response.  
2003

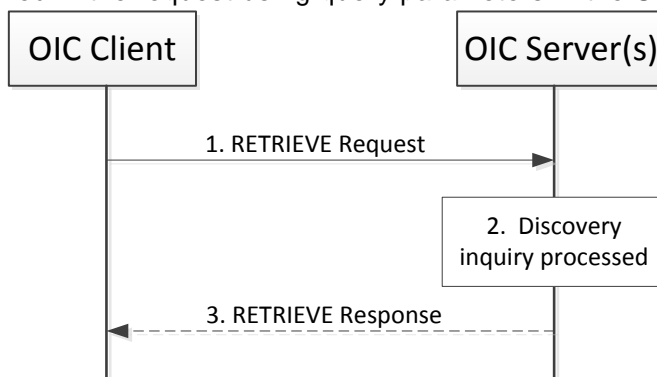


2004  
2005

**Figure 27. Resource based discovery: Information publication process**

2006 **11.3.4 Resource based discovery: Finding information**

2007 The discovery process (Figure 28) is initiated as a RETRIEVE request to the resource to enable  
 2008 discovery. The request may be sent to a single Device (as in a Unicast) or to multiple Devices (as  
 2009 in Multicast). The specific mechanisms used to do Unicast or Multicast are determined by the  
 2010 support in the data connectivity layer. The response to the request has the information to be  
 2011 discovered based on the policies for that information. The policies can determine which information  
 2012 is shared, when and to which requesting agent. The information that can be discovered can be  
 2013 resources, types, configuration and many other standards or custom aspects depending on the  
 2014 request to appropriate resource and the form of request. Optionally the requester may narrow the  
 2015 information to be returned in the request using query parameters in the URI query.



2016  
2017

**Figure 28. Resource based discovery: Finding information**

2018

2019 **Discovery Resources**

2020 Some of the Core Resources shall be implemented on all Devices to support discovery. The Core  
 2021 Resources that shall be implemented to support discovery are:

- 2022 ● /oic/res for discovery of resources
- 2023 ● /oic/p for discovery of platform
- 2024 ● /oic/d for discovery of device information

2025 Details for these mandatory Core Resources are described in Table 14

2026 Platform resource –

2027 The OCF recognizes that more than one instance of Device may be hosted on a single platform.  
 2028 Clients need a way to discover and access the information on the platform. The core resource,  
 2029 /oic/p exposes platform specific properties. All instances of Device on the same Platform shall  
 2030 have the same values of any properties exposed (i.e. an Device may choose to expose optional  
 2031 properties within /oic/p but when exposed the value of that property should be the same as the  
 2032 value of that property on all other Devices on that Platform)  
 2033

2034 **Device resource**

2035 The device resource shall have the pre-defined URI /oic/d. The resource /oic/d exposes the  
 2036 properties pertaining to a Device as defined in Table 14. The properties exposed are determined  
 2037 by the specific instance of Device and defined by the resource type(s) of /oic/d on that Device.  
 2038 Since all the resource types of /oic/d are not known a priori, the resource type(s) of /oic/d shall be  
 2039 determined by discovery through the core resource /oic/res. The device resource /oic/d shall have  
 2040 a default resource type that helps in bootstrapping the interactions with this device (the default  
 2041 type is described in Table 14.)  
 2042

2043 **Protocol indication**

2044 A Device may need to support different messaging protocols depending on requirements for  
 2045 different application profiles. For example, the Smart Home profile may use CoAP and the  
 2046 Industrial profile may use DDS. To enable interoperability, a Device uses the protocol indication  
 2047 to indicate the transport protocols they support and can communicate over.  
 2048

2049 **Table 14. Mandatory discovery Core Resources**

| Pre-define d URI | Resource Type Title | Resource Type ID ("rt" value)                                 | Interfaces | Description   | Related Functional Interaction |
|------------------|---------------------|---|------------|---|--------------------------------|
| /oic/res         | Default             | oic.wk.res  | oic.if.ll  | The resource through which the corresponding Server is discovered and introspected for available resources.<br><br>/oic/res shall expose the resources that are discoverable on a Device. When an Server receives a RETRIEVE request targeting /oic/res (e.g., GET /oic/res), it shall respond with the link list of all the discoverable resources of itself. The /oic/d and /oic/p are discoverable resources, hence their links are included in /oic/res response. The resource properties exposed by /oic/res are listed in Table 15.   | Discovery                      |
| /oic/p           | Platform            | oic.wk.p  | oic.if.r   | The discoverable resource through which platform specific information is discovered.<br><br>The <b>resource properties</b> exposed by /oic/p are listed in Table 17   | Discovery                      |
| /oic/d           | Device              | oic.wk.d and/or one or more Device Specific resource type IDs | oic.if.r   | The discoverable via /oic/res resource which exposes properties specific to the Device instance.<br><br>The <b>resource properties</b> exposed by /oic/d are listed in Table 17<br><br>/oic/d may have one or more resource types that are specific to Device in addition to the default resource type or if present overriding the default resource type.<br><br>The base type oic.wk.d defines the properties that shall be exposed by all Devices.<br><br>The device specific resource type(s) exposed are dependent on the class of device (e.g. air conditioner, smoke alarm); applicable values are defined by the vertical specifications. | Discovery                      |

2050

2051 Table 15 defines oic.wk.res resource type.

2052

**Table 15. oic.wk.res resource type definition**

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

2053 A Device which wants to indicate its messaging protocol capabilities may add the property 'mpro'  
 2054 in response to a request on /oic/res. A Device shall support CoAP based discovery as the baseline  
 2055 discovery mechanism (see section 10.2). A Client which sees this property in a discovery response  
 2056 can choose any of the supported messaging protocols for communicating with the Server for further  
 2057 messages. For example, if a Device supporting multiple protocols indicates it supports a value of  
 2058 "1 3" for the 'mpro' property in the discovery response, then it cannot be assumed that there is an  
 2059 implied ordering or priority. But a vertical service specification may choose to specify an implied  
 2060 ordering or priority. If the 'mpro' property is not present in the response, A Client shall use the  
 2061 default messaging protocol as specified in the vertical specification for further communication.  
 2062 Table 16 provides an OCF registry for protocol schemes.

2063

**Table 16. Protocol scheme registry**

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

2064 Note: The discovery of an endpoint used by a specific protocol is out of scope. The mechanism used by a Client to form  
 2065 requests in a different messaging protocol other than discovery is out of scope.

2066

2067 The following applies to the use of /oic/d as defined above:

2068 • A vertical may choose to expose its Device Type (e.g., refrigerator or A/C) by adding the Device  
 2069 Type to the list of Resource Types associated with /oic/d.

2070 ○ For example; rt of /oic/d becomes ["oic.wk.d", "oic.d.<thing>"]; where "oic.d.<thing>"  
 2071 is defined in another spec such as the Smart Home vertical.

2072 ○ This implies that the properties exposed by /oic/d are by default the mandatory  
 2073 properties in Table 17.

2074 • A vertical may choose to extend the list of properties defined by the Resource Type 'oic.wk.d'.  
 2075 In that case, the vertical shall assign a new Device Type specific Resource Type ID. The  
 2076 mandatory properties defined in Table 17 shall always be present.

2077 Note:

2078 As per existing Core specification definitions the resource type ID may be a list of resource type IDs; when that is the  
 2079 case the default resource type ID for /oic/d is the first resource type ID listed. So a vertical can list 'oic.d.thing' first.  
 2080 This then means a GET /oic/d returns the properties for oic.d.thing and a GET /oic/d?rt=<some rt> returns the properties  
 2081 for the rt listed in the query.

2082 Table 17 oic.wk.d resource type definition defines the base resource type for the /oic/d resource.  
 2083

2084

**Table 17. oic.wk.d resource type definition**

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

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

2085

2086 The additional resource type(s) of the /oic/d resource are defined by the vertical specification.

2087

2088 Table 18 defines oic.wk.p resource type.

2089

2090

**Table 18. oic.wk.p resource type definition**

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



|                  |     |        |  |  |   |    |  |
|------------------|-----|--------|--|--|---|----|--|
|                  |     |        |  |  |   |    | with the "T" as a delimiter between "date" and "time". The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.           |
| <b>Vendor ID</b> | vid | string |  |  | R | no | Vendor defined string for the platform. The string is freeform and up to the vendor on what text to populate it. |

2091

2092 **Composite Device**

2093 A physical device may be modelled as a single device or as a composition of other devices. For  
 2094 example a refrigerator may be modelled as a composition, as such part of its definition of may  
 2095 include a sub-tending thermostat device which itself may be composed of a sub-tending  
 2096 thermometer device.

2097 There may be more than one way to model an server as a composition. One example method  
 2098 would be to have Platform which represents the composite device to have more than one instance  
 2099 of a Device on the Platform. Each Device instance represents one of the distinct devices in the  
 2100 composition. Each instance of Device may itself have or host multiple instances of other resources.

2101 An implementation irrespective of how it is composed shall only expose a single instance of /oic/d  
 2102 with an 'rt' of choice for each logical Server.

2103 Thus, for the above refrigerator example if modeled as a single Server; /oic/res would expose  
 2104 /oic/d with a resource type name appropriate to a refrigerator. The sub-tending thermostat and  
 2105 thermometer devices would be exposed simply as instances of a resource with a device  
 2106 appropriate resource type with an associated URI assigned by the implementation; e.g.,  
 2107 /MyHost/MyRefrigerator/Thermostat and /MyHost/MyRefrigerator/Thermostat/Thermometer.

2108

2109 **11.3.5 Resource discovery using /oic/res**

2110 Discovery using /oic/res is the default discovery mechanism that shall be supported by all Devices  
 2111 as follows:

- 2112 a) Every Device updates its local /oic/res with the resources that are discoverable (see section  
 2113 7.3.2.2). Every time a new resource is instantiated on the Device and if that resource is  
 2114 discoverable by a remote Device then that resource is published with the /oic/res resource that  
 2115 is local to the Device (as the instantiated resource).
- 2116 b) An Device wanting to discover resources or resource types on one or more remote Devices  
 2117 makes a RETRIEVE request to the /oic/res on the remote Devices. This request may be sent  
 2118 multicast (default) or unicast if only a specific host is to be probed. The RETRIEVE request  
 2119 may optionally be restricted using appropriate clauses in the query portion of the request.  
 2120 Queries may select based on resource types, interfaces, or properties.
- 2121 c) Query applies to the representation of the resources. /oic/res is the only resource whose  
 2122 representation has "rt". So /oic/res is the only resource that can be used for Multicast discovery  
 2123 at the transport protocol layer.
- 2124 d) The Device receiving the RETRIEVE request responds with a list of resources, the resource  
 2125 type of each of the resources and the interfaces that each resource supports. Additionally

2126 information on the policies active on the resource can also be sent. The policy supported  
2127 includes observability and discoverability. (More details below)

2128 e) The receiving Device may do a deeper discovery based on the resources returned in the  
2129 request to /oic/res.

2130

2131 The information that is returned on discovery against /oic/res is at the minimum:

- 2132 • The URI (relative or fully qualified URL) of the resource
- 2133 • The Resource Type of each resource. More than one Resource Type may be returned if the  
2134 resource enables more than one type. To access resources of multiple types, the specific  
2135 resource type that is targeted shall be specified in the request.
- 2136 • The Interfaces supported by that Resource. Multiple interfaces may be returned. To access a  
2137 specific interface that interface shall be specified in the request. If the interface is not specified,  
2138 then the Default Interface is assumed.
- 2139 • Policies defined against that resource. These policies may be security related, access modes,  
2140 types of interactions, etc. In addition to the request/response type of interaction, the  
2141 specification allows the resource to be “observed” (section 11.4.2).

2142

2143 The JSON schemas for discovery using /oic/res are described in D.8. Also refer to Section 10  
2144 (Endpoint Discovery) for details of Multicast discovery using /oic/res on a CoAP transport.

2145 After performing discovery using /oic/res, Clients may discover additional details about Server by  
2146 performing discovery using /oic/p, /oic/rts etc. If a Client already knows about Server it may  
2147 discover using other resources without going through the discovery of /oic/res.

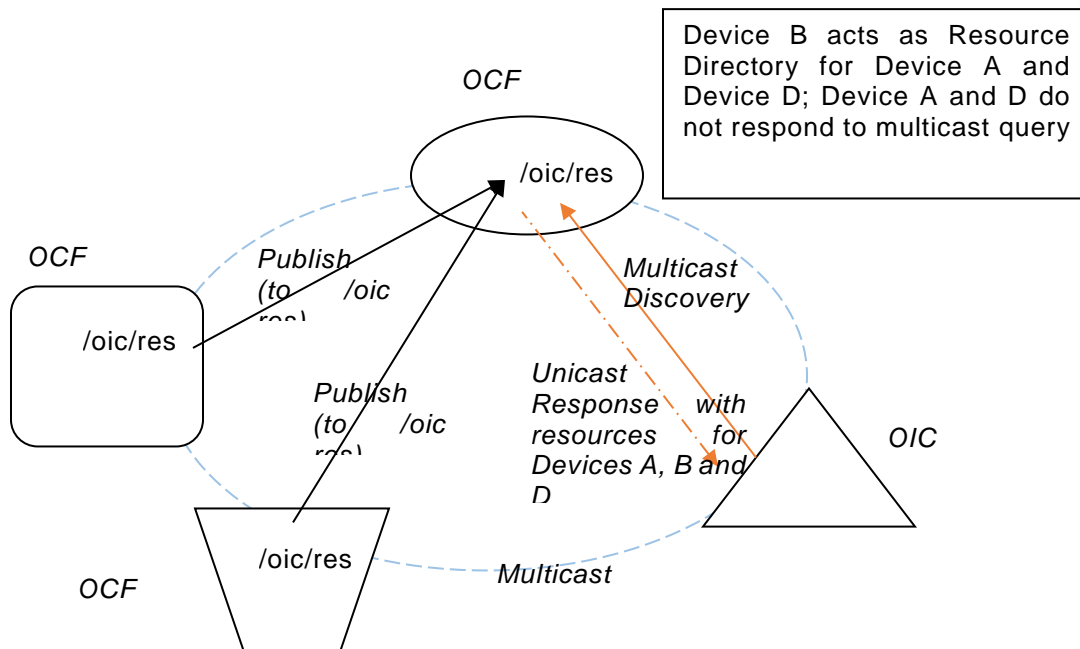
## 2148 **11.3.6 Resource directory (RD) based discovery**

### 2149 **11.3.6.1 Introduction**

#### 2150 **11.3.6.1.1 Indirect discovery for lookup of the resources**

2151 Direct discovery is the mechanism used currently to find resources in the network. When needed,  
2152 resources are queried at a particular node directly or a multicast packet is sent to all nodes. Each  
2153 queried node responds directly with its discoverable resources to the discovering device.  
2154 Resources available locally are registered on the same device.

2155 In some situations, one of the other mechanisms described in section 11.3.2.3, called indirect  
2156 discovery, may be required. Indirect discovery is when a 3rd party device, other than the  
2157 discovering device and the discovered device, assists with the discovery process. The 3rd party  
2158 only provides information on resources on behalf of another device but does not host resources  
2159 on part of that device.



2160

2161

**Figure 29. Indirect discovery of resource by resource directory**

2162 Indirect discovery is useful for a resource constrained device that needs to sleep to manage power  
 2163 and cannot process every discovery request, or when devices may not be on the same network  
 2164 and requires optimization for discovery. Once resources are discovered using indirect discovery  
 2165 then the access to the resource is done by a request directly to the Device that hosts that resource.

### 2166 11.3.6.1.2 Resource directory

2167 A resource directory (RD) is an Device that assists with indirect discovery. A RD can be queried  
 2168 at its /oic/res resource to find resources hosted on other Devices. These Devices can be sleepy  
 2169 nodes or any other device that cannot or may not respond to discovery requests. Device can  
 2170 publish all or partial list of resources they host to a RD. The RD then responds to queries for  
 2171 Resource discovery on behalf of the publishing Device (for example: when a Device may go to  
 2172 sleep). For general Resource discovery, the RD behaves like any other Server in responding to  
 2173 requests to /oic/res.

2174 Any Device that serves or acts as a RD shall expose a well-known resource /oic/rd. The Devices  
 2175 that want to discover RDs shall use this resource and one of the Resource discovery mechanisms  
 2176 to discover the RD and get the parameters of the RD. The information discovered through this  
 2177 resource shall be used to select the appropriate RD to use for resource publication. The bias  
 2178 information shall include the following criteria: power source (AC, battery powered or safe/reliable),  
 2179 connectivity (wireless, wired), CPU, memory, load statistics (processing publishing and query from  
 2180 the devices). In addition, the RD shall return a bias factor that ranges from 0 to 100. Optionally,  
 2181 the RD may also return a context - the value which shall be a string and semantics of the context  
 2182 are not discussed in this document but it is expected that the context will be used to establish a  
 2183 domain, region or some such scope that is meaningful to the application, deployment or usage.

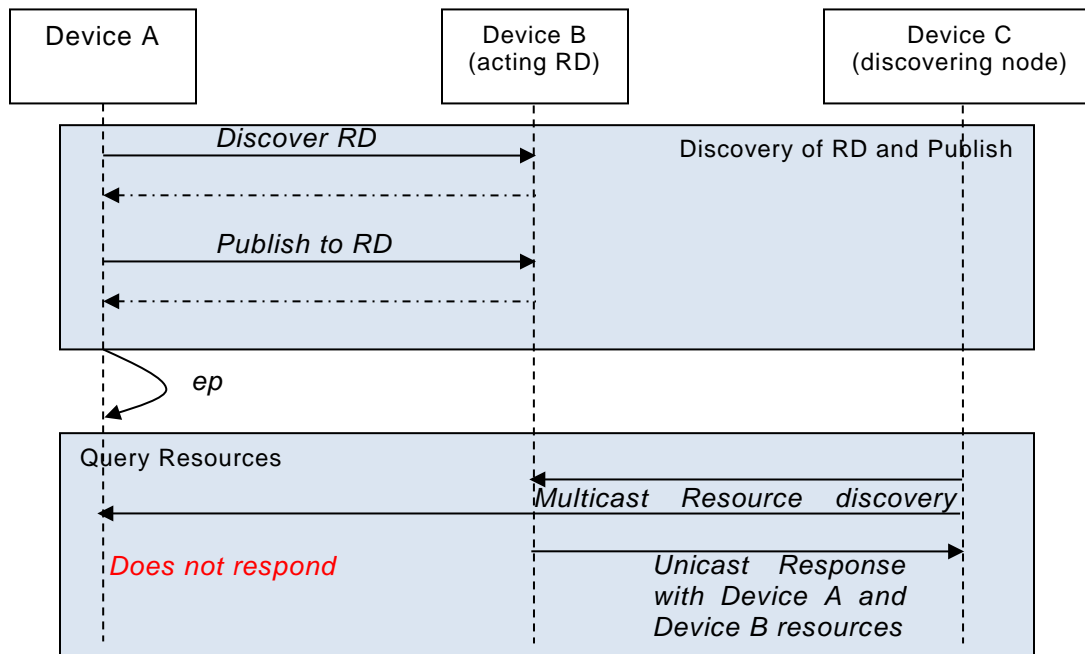
2184 Using these criteria or the bias factor, the Device shall select one RD (per context) to publish its  
 2185 resources. A context describes the state of an OCF Device with respect to Resource discovery. A  
 2186 context is usually determined at deployment and from application requirements. An example of a  
 2187 context could be a multicast group- a Device that is a member of more than one multicast group  
 2188 may have to find and select a RD in each of the multicast groups (i.e. per context) to publish its  
 2189 information. The Device may choose other RDs during its lifetime but a Device shall not publish

2190 its resource information to more than one RD Devices such as TV, network router, desktop will  
2191 have higher weightage or bias factor compared to mobile phone device.

2192 **11.3.6.2 The remainder of this section is divided into two parts. The first part covers**  
2193 **discovering of the RD and publishing, updating and deleting of resources for**  
2194 **the constrained/sleepy device. The second part covers the replies of the RD to**  
2195 **queries from devices with the aim to discover resources. Resource directory**  
2196 **discovery**

2197 **11.3.6.2.1 Discovering a resource directory**

2198 A RD in the OCF network shall support RD discovery, shall provide the facility to allow devices to  
2199 publish their resource information to a RD, to update resource information in a RD and to delete  
2200 resource information from a RD.



2201

2202

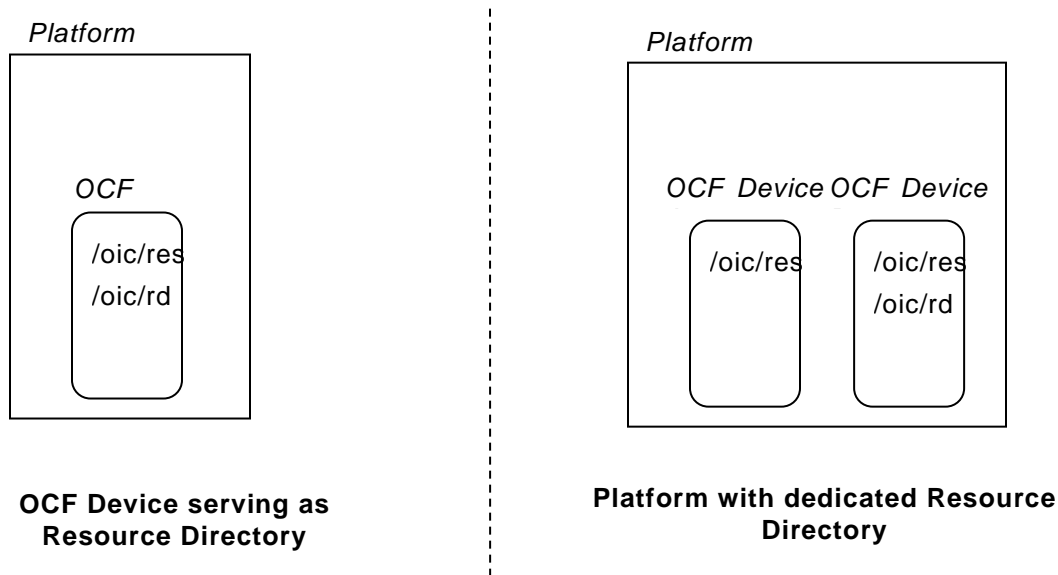
**Figure 30. RD discovery and RD supported query of resources support**

2203 As shown in Figure 30, the Device that wishes to advertise its resources: first discovers a resource  
2204 directory and then publishes the desired resource information. Once a set of resources have been  
2205 published to a RD then the publishing device shall not respond to multicast Resource discovery  
2206 queries for those published resources when the RD is on the same multicast domain. In that case,  
2207 only the RD shall respond to multicast Resource discovery requests on the resource published to  
2208 it.

2209 An OCF network allows for more than one device acting as a RD. The reason to have multiple RD  
2210 support is to make network scalable, handle network failures and centralized device failure  
2211 bottleneck. This does not preclude a scenario where a use case or deployment environment may  
2212 require single device in the environment to be deployed as the only resource directory (e.g.  
2213 gateway model). There may be more than one Device acting as RD on a Platform.

2214 Discovering of an RD may result in responses from more than one RD. The discovering device  
2215 shall select a RD. The selection may be based on the weightage parameter(s) provided in the  
2216 response from the RD.

2217 An RD will be application agnostic i.e., application should not be aware whether resource directory  
 2218 was queried to get the resource information. All the handling of the retrieval is kept opaque to the  
 2219 application. A Client that performs Resource discovery uses an RD just like it may use any other  
 2220 Server for discovery. It may send a unicast request to the RD when it needs only the resource  
 2221 advertised on the RD or do a multicast query when it does not require or have explicit knowledge  
 2222 of an RD.



2223  
 2224 **Figure 31. Resource Direction Deployment Scenarios**

2225 Resource directory can also be discovered in the following manners:

- 2226 • Pre-configuration: Devices wishing to publish resource information may be configured a priori  
 2227 with the information (e.g. IP address, port, transport etc.) of a specific resource directory. This  
 2228 pre-configuration may be done at onboarding or may be updated on the device using an out-  
 2229 of-band method. This pre-configuration may be done by the manufacturer or by the user/device  
 2230 manager.
- 2231 • Query-oriented: A Client wanting to discover resource directories using query-oriented  
 2232 discovery (i.e. pull) shall issue multicast Resource discovery request directed to the /oic/rd  
 2233 resource. Only the devices that hosts a /oic/rd resource shall respond to this query. The  
 2234 response shall include information about the RD (as defined by the resource type) and  
 2235 weightage parameters to allow the discovering device to select between RDs (see details in  
 2236 RD selection section). The /oid/rd resource shall be instantiated on the OCF Devices acting as  
 2237 a resource directory. The /oic/rd schema is as defined in D.13.
- 2238 • Advertisement: An RD may advertise about itself to devices. It is an advertisement packet. The  
 2239 devices that are already publishing to a RD may use this as a heartbeat message of the RD. If  
 2240 the RD advertisement does not arrive at a stipulated interval, publishing device starts searching  
 2241 for other RDs in the network, as this is a signal that RD is not online. Other usage of this  
 2242 message is it serves as an advertisement for a device seeking a RD to publish their resources.  
 2243 The details from the advertisement can then be used to query directly to a RD to get weightage  
 2244 details instead of sending a multicast packet in a network. As it is intended this is sent at a  
 2245 regular interval and does not include weightage information to keep packet sizes small.
- 2246 • One of the important benefits of an RD is to make services discoverable in networks that don't  
 2247 support site wide multicast but do support site wide routing. An example of such a network is  
 2248 Homenet .To enable an RD function across such a network a site discovery mechanism is  
 2249 needed to discover the RD service (IP address & port number). Homenets that support hybrid

2250 proxy (IETF draft-ietf-homenet-hybrid-proxy-zeroconf-00) allow site wide discovery based on  
 2251 dns-sd/mDNS. In order to make itself discoverable beyond the link local scope, an RD with a  
 2252 routable ip address shall implement the mDNS responder requirements defined in  
 2253 IETF RFC 6762. The RD shall respond to mDNS queries of type PTR and with a service name  
 2254 equal to "\_rd.\_sub.\_oic.\_udp.local". The response shall include all routable IP addresses.  
 2255 Devices with a routable ip address shall discover all available RD instances by issuing a DNS-  
 2256 SD's PTR lookup as defined in IETF RFC 6763 with as service name service name  
 2257 "\_rd.\_sub.\_oic.\_udp.local". The response shall include all routable addresses/port pair through  
 2258 which the RD service is made accessible.

2259 **11.3.6.2.2 Resource directory selection process**

2260 **11.3.6.2.2.1 Selection criteria**

2261 When a device discovers more than one RD then it shall decide to use one of these RDs based on  
 2262 the selection criteria described here. A device shall use or publish information to only one RD  
 2263 within a multicast domain at a given time. This is to minimize the burden of processing duplicate  
 2264 information in the Resource discovery phase.

2265 There two ways to select an RD. One is based on a bias factor (RD generated) and the other is  
 2266 based on clients determination based on granular parameters provided by the server (client/device  
 2267 generated). Devices may use one or both methods to select an RD.

2268 *Bias factor:* The bias factor is a server generated positive number in the range of 0 to 100, where  
 2269 0 is the lowest to 100 being the highest. If two RDs have the same bias factor then the selecting  
 2270 device may choose either based auxiliary criteria or at random. Either way only one RD shall be  
 2271 selected and used at a time. No specific method is defined in this specification to determine the  
 2272 bias factor for an RD. The number may be a pre-configured value at the time of onboarding or  
 2273 subsequent configuration of the RD or may be based on a formula determined by the  
 2274 implementation of the RD. (OCF will provide a standard formula for this calculation in a future  
 2275 version or release of specification).

2276 The bias factor shall be calculated by the RD by adding the contribution values determined for  
 2277 each of the parameters in Table 19 and divided by the number of parameters. An RD may advertise  
 2278 a bias factor larger than the calculated value when there is reason to believe that the RD is highly  
 2279 capable for example an installed service provider gateway.

2280 *Parameters:* Optionally, parameters defined in Table 19 (like direct power supply, network  
 2281 connectivity, load conditions, CPU power, memory, etc.) may be returned in the discovery  
 2282 response. Discovering device may use the details to make granular selection decisions based on  
 2283 client defined policies and criteria that use the RD parameters. For example, a device in an  
 2284 industrial deployment may not weight power connectivity high but another in home environments  
 2285 may give more weightage for power.

2286 **Table 19: Selection parameters**

| Parameter | Values (Contribution)              | Description  |
|-----------|------------------------------------|--|
| Power     | Safe (100)<br>AC (70)<br>Batt (40) | <ul style="list-style-type: none"> <li>Safe implies that the power supply is reliable and is backed up with battery for power outages etc.</li> <li>Implementation may lower the number for Batt based on the type of battery the RD device runs on. If battery conservation is important then this number should be lowered.</li> </ul> |
| Mobility  | Fixed (100)<br>Mobile (50)         | <ul style="list-style-type: none"> <li>Implementation may further grade the mobility number based on how mobile the RD device is; lower number for highly mobile and larger numbers for limited mobility</li> <li>The mobility number shall not be larger than 80</li> </ul>   |

|                     |  |   |
|---------------------|--|---|
| Network Product     | Type:<br><ul style="list-style-type: none"> <li>Wired (10)</li> <li>Wireless (4)</li> </ul> Bandwidth:<br><ul style="list-style-type: none"> <li>High (10)</li> <li>Low (5)</li> <li>Lossy (3)</li> </ul> Interfaces | <ul style="list-style-type: none"> <li>Network product = [sum of (type * bandwidth per network interface)]/[number of interfaces]</li> <li>Normalized to 100</li> </ul>   |
| Memory Factor       | Available<br>Total   | <ul style="list-style-type: none"> <li>Memory is the volatile or non-volatile storage used to store the resource information</li> <li>Memory Factor = [Available]/[Total]</li> <li>Normalized to 100 (i.e. expressed as percentage)</li> </ul>  |
| Request Load Factor | 1-minute<br>5-minute<br>15-minutes   | <ul style="list-style-type: none"> <li>Current request loading of the RD</li> <li>Similar to UNIX load factor (using observable, pending and processing requests instead of runnable processes)</li> <li>Expressed as a load factor 3-tuple (up to two decimal points each). Factor is based on request processed in a 1-minute (L1), 5-minute (L5) and 15-minute (L15) windows</li> <li>See <a href="http://www.teamquest.com/import/pdfs/whitepaper/ldavg1.pdf">http://www.teamquest.com/import/pdfs/whitepaper/ldavg1.pdf</a></li> <li>Factor = <math>100 - ((L1*3 + L5*7 + L15*10)/3)</math></li> </ul> |

2287

2288 **11.3.6.2.2.2 Selection scenarios**

2289 The device that wants to use an RD will use the endpoint discovery to find zero or more RDs on  
2290 the network. After discovering the RDs, the device needs to select an RD of all found RDs on the  
2291 network. The selection based on the bias factor will ensure that an Device can judge if the found  
2292 RD is suitable for its needs.

2293 The following situation can occur during the selection of an RD:

- 2294 1) A single or multiple RDs are present in the network
- 2295 2) No RD is present in the network
- 2296 3) an additional RD arrives on the network

2297

2298 In the first scenario the RDs are already present. If a single RD is detected then that RD can be  
2299 used . When multiple RDs are detected the Device uses the bias information to select the RD.

2300

2301 In the second scenario, device will listen to the advertisement of the devices that hosts the RDs.  
2302 Once an RD advertisement packet is received it judges if the bias criteria are met and starts using  
2303 the RDs.

2304

2305 In the third scenario the Device has already published its resources to an existing RD. In this  
2306 scenario it discovers a new RD on the network.

2307 After judging the bias factor the Device may choose to move to the new RD.

2308

2309 **11.3.6.3 If the decision is made to select the new RD, the then Device shall delete its**  
2310 **resource information from the current used RD and then after removal publish**  
2311 **the information to the new RD. During the transition period the Device itself**  
2312 **shall respond to Resource discovery requests. Resource publishing**

#### 2313 **11.3.6.3.1 Publish resources**

##### 2314 **11.3.6.3.1.1 Overview**

2315 After the selection process of a RD, a device may choose one of the following mechanisms:

- 2316 • Push its resources information to the selected RD or
- 2317 • Request the RD to pull the resource information by doing a unicast discovery request against  
2318 its /oic/res

2319 The publishing device may decide to publish all resources or few resources on the resource  
2320 directory. The publishing device shall only publish resources that are otherwise published to its  
2321 own /oic/res. A publishing device may respond to discovery requests (on its /oic/res resource) for  
2322 the resources it does not publish to a RD. Nonetheless, it is highly recommended that when an RD  
2323 is used, all discoverable resources on the publisher be published to the RD.

##### 2324 **11.3.6.3.1.2 Publish: Push resource information**

2325 Resource information is published using an UPDATE CRUDN operation to /oic/rd using the  
2326 resource type oic.wk.rdpub and the oic.if.baseline interface.

2327 Once a publishing device has published resources to a RD, it may not respond to the multicast  
2328 discovery queries for the same resources against its own /oic/res, especially when on the same  
2329 multicast domain as the RD. After publishing resources, it is a RD responsibility to reply to the  
2330 queries for the published resources.

2331 If the publishing device is in sleep mode and a RD has replied on behalf of the publishing device,  
2332 then a discovering device will try to access resource on the provided URI.

2333 There is another possibility that the resource directory and the publishing device both respond to  
2334 the multicast query from the discovering device. This will create a duplication of the packet but is  
2335 an alternate that may be used for non-robust network. It is not a recommended option but for  
2336 industrial scenarios, this is one of the possibilities. Either way, discovering clients shall always be  
2337 prepared to process duplicate information in responses to multicast discovery request. The /oic/rd  
2338 schema is as defined in D.13 to specify publishing (oic.rd.publish) to the /oic/rd resource.

##### 2339 **11.3.6.3.2 Update resource information**

2340 Server will hold the publish resource information till the time specified in the ttl field. A device can  
2341 send update if it seeks a RD to keep holding resources and reply to queries on its behalf. Update  
2342 can be used for updating about all resources that are published on a RD or can use to do per  
2343 resource published.

2344 Updates are done using the same resource type and interface as for the initial publish but only the  
2345 information to be updated is provided in the payload.

##### 2346 **11.3.6.3.3 Delete resource information**

2347 A resource information hold at the resource directory can be removed anytime by the publishing  
2348 device. It can be either for the whole device information or for a particular resource. This resource  
2349 should be only allowed when device meets a certain requirement, as it can create potential security  
2350 issue.



2351 The delete is done using the device ID “id” as the tag in DELETE request query when all the  
2352 resource information from the device is to be deleted. In the case of a specific resource then the  
2353 DELETE request shall include the instance “ins” tag along with the device ID in the query.

2354 Selective deletion of information for individual resources is not possible the case where the RD  
2355 pull the resource information. The publishing device can request a delete but only for all the  
2356 resource information that the RD has pulled from that device. In this case, the DELETE request  
2357 has the device ID “id” tag in the query.

#### 2358 **11.3.6.3.4 Transfer resource information from one RD to another**

2359 When a publishing device identifies an RD that is better suited, it may decide to publish to that RD.  
2360 Since the device shall publish to only one RD at a time, the client shall ensure that previously  
2361 published information is deleted from the currently used RD before publishing to the newly selected  
2362 RD. The deletion of the resource may be done either by allowing the TTL to expire or explicitly  
2363 deleting the resource information.

2364 RDs shall not communicate resource information between themselves. It is the client’s  
2365 responsibility to choose the RD and to manage the published resources.

#### 2366 **11.3.6.4 Resource discovery**

##### 2367 **11.3.6.4.1 Query and retrieving of the resources**

2368 The query based discovery process remains the same as that in the absence of an RD. Resources  
2369 may be discovered by querying the /oic/res resource by sending a multicast or unicast request. In  
2370 the case of a multicast discovery request, an RD will respond for the device that hosts the  
2371 resources. Clients shall be prepared to process duplicate resource information from more than one  
2372 RD responding with the same information or from an RD and the hosting device (publishing the  
2373 resource information) both responding to the request. Interaction with resources discovered using  
2374 the RD is done using the same mechanism and methods as with resources discovered by querying  
2375 the /oic/res resource of the device hosting the resources (e.g., connect to the resource and perform  
2376 CRUDN operations on the resource).

#### 2377 **11.4 Notification**

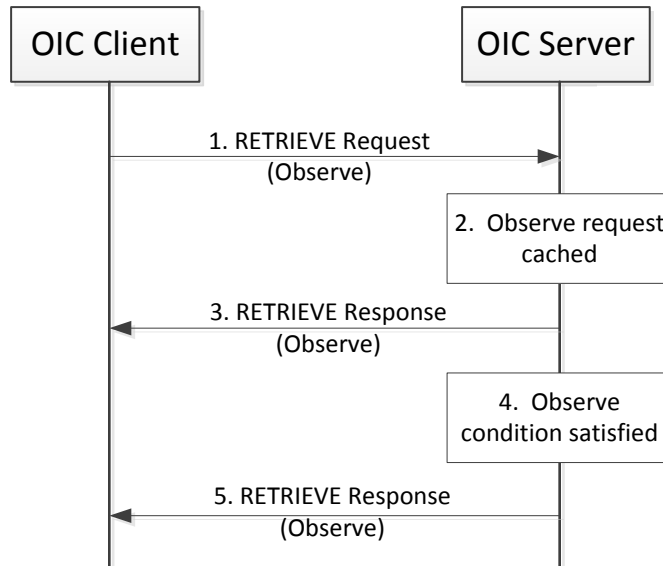
##### 2378 **11.4.1 Overview**

2379 An Server shall support NOTIFY operation to enable a Client to request and be notified of desired  
2380 states of one or more Resources in an asynchronous manner. Section 11.4.2 specifies the observe  
2381 mechanism in which updates are delivered to the requester.

##### 2382 **11.4.2 Observe**

2383 In observe mechanism the Client utilizes the RETRIEVE operation to require the Server for updates  
2384 in case of Resource state changes. The Observe mechanism consists of five steps which are  
2385 depicted in Figure 32 and described below.

2386 Note: the observe mechanism can only be used for a resource with a property of observable  
2387 (section 7.3.2.2).



2388

2389

**Figure 32. Observe Mechanism**

2390 **11.4.2.1 RETRIEVE request with observe indication**

2391 The Client transmits a RETRIEVE request message to the Server to request updates for the  
 2392 Resource on the Server if there is a state change. The RETRIEVE request message carries the  
 2393 following parameters:

- 2394 • *fr*: Unique identifier of the Client
- 2395 • *to*: Resource that the Client is requesting to observe
- 2396 • *ri*: Identifier of the RETRIEVE request
- 2397 • *op*: RETRIEVE
- 2398 • *obs*: Indication for observe request

2399 **11.4.2.2 Processing by the Server**

2400 Following the receipt of the RETRIEVE request, the Server may validate if the Client has the  
 2401 appropriate rights for the requested operation and the properties are readable and observable. If  
 2402 the validation is successful, the Server caches the information related to the observe request. The  
 2403 Server caches the value of the *ri* parameter from the RETRIEVE request for use in the initial  
 2404 response and future responses in case of a change of state.

2405 **11.4.2.3 RETRIEVE response with observe indication**

2406 The Server shall transmit a RETRIEVE response message in response to a RETRIEVE request  
 2407 message from a Client. The RETRIEVE response message shall include the following parameters.  
 2408 If validation succeeded, the response includes an observe indication. If not, the observe indication  
 2409 is omitted from the response which signals to the requesting client that registration for notification  
 2410 was not allowed.

2411 The RETRIEVE response message shall include the following parameters:

- 2412 • *fr*: Unique identifier of the Server
- 2413 • *to*: Unique identifier of the Client
- 2414 • *ri*: Identifier included in the RETRIEVE request
- 2415 • *cn*: Information resource representation as requested by the Client

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

2418 **11.4.2.4 Resource monitoring by the Server**

2419 The Server shall monitor the state the Resource identified in the observe request from the Client.  
 2420 Anytime there is a change in the state of the observed resource, the Server sends another  
 2421 RETRIEVE response with the observe indication. The mechanism does not allow the client to  
 2422 specify any bounds or limits which trigger a notification, the decision is left entirely to the server.

2423 **11.4.2.5 Additional RETRIEVE responses with observe indication**

2424 The Server shall transmit updated RETRIEVE response messages following observed changes in  
 2425 the state of the Resources indicated by the Client. The RETRIEVE response message shall include  
 2426 the parameters listed in section 11.4.2.3.

2427 **11.4.2.6 Cancelling Observe**

2428 The Client can explicitly cancel observe by sending a RETRIEVE request without the observe  
 2429 indication field to the same resource on Server which it was observing. For certain protocol  
 2430 mappings, the client may also be able to cancel an observe by ceasing to respond to the  
 2431 RETRIEVE responses.

2432 **11.5 Device management**

2433 The Device Management includes the following functions:

- Diagnostics and maintenance

2435 The device management functionalities specified in this version of specification are intended to  
 2436 address the basic device management features. Addition of new device management features in  
 2437 the future versions of the specification is expected.

2438 **11.5.1 Diagnostics and maintenance**

2439 The Diagnostics and Maintenance function in the Framework is intended for use by the  
 2440 administrators to resolve issues encountered with the Devices while operating in the field. If  
 2441 diagnostics and maintenance is supported by a Device, the Core Resource '/oic/mnt' shall be  
 2442 supported as described in Table 20.

2443 **Table 20. Optional diagnostics and maintenance device management Core Resources**

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

2444  
 2445 Table 21 defines the oic.wk.mnt resource type. At least one of the Factory\_Reset, and Reboot  
 2446 properties shall be implemented.

2447 **Table 21. oic.wk.mnt resource type definition**

| Property title | Property name | Value type | Value rule | Unit | Access mode | Mandatory | Description |
|----------------|---------------|------------|------------|------|-------------|-----------|-------------|
| <b>Name</b>    | n             | string     |            |      | R, W        | no        |             |

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

2448

2449 Note: \* - Default indicates the value of this property as soon as the device is rebooted or factory reset

2450

2451 The Framework specifies the following commands to be executed on the designated diagnostic  
2452 resource of Devices over the network:

- 2453 • Factory\_Reset: Updates the device configuration to its original (default) state (factory state  
2454 and equivalent to hard reboot)
- 2455 • Reboot: Triggers a soft reboot of a Device maintaining most of the configurations intact

2456 Execution of these commands may result in a change in the configuration state of a Device. The  
2457 configuration information in the configuration resource is expected to be updated following  
2458 execution of these commands by the Device, if needed. A Client invokes operations on the Server  
2459 for executing the Diagnostic functions by sending an UPDATE message to the Server.

2460

## 2461 11.6 Scenes

### 2462 11.6.1 Introduction

2463 Scenes are a mechanism for automating certain operations.

2464 A scene is a static entity that stores a set of defined resource property values for a collection of  
2465 resources. Scenes provide a mechanism to store a setting over multiple Resources that may be  
2466 hosted by multiple separate Servers. Scenes, once set up, can be used by multiple Clients to recall  
2467 a setup.

2468 Scenes can be grouped and reused, a group of scenes is also a scene.

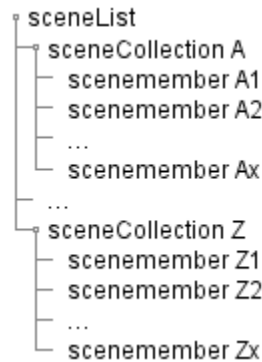
2469 In short, scenes are bundled user settings.

2470 **11.6.2 Scenes**

2471 **11.6.2.1 Introduction**

2472 Scenes are described by means of resources. The scene resources are hosted by a Server and  
2473 the top level resource is listed in /oic/res. This means that a Client can determine if the scene  
2474 functionality is hosted on a Server via a RETRIEVE on /oic/res or via Resource discovery. The  
2475 setup of scenes is driven by Client interactions. This includes creating new scenes, and mappings  
2476 of Server resource properties that are part of a scene.

2477 The scene functionality is created by multiple resources and has the structure depicted in Figure  
2478 33. The sceneList and sceneCollection resources are overloaded collection resources. The  
2479 sceneCollection contains a list of scenes. This list contains zero or more scenes. The  
2480 sceneMember resource contains the mapping between a scene and what needs to happen  
2481 according to that scene on an indicated resource.

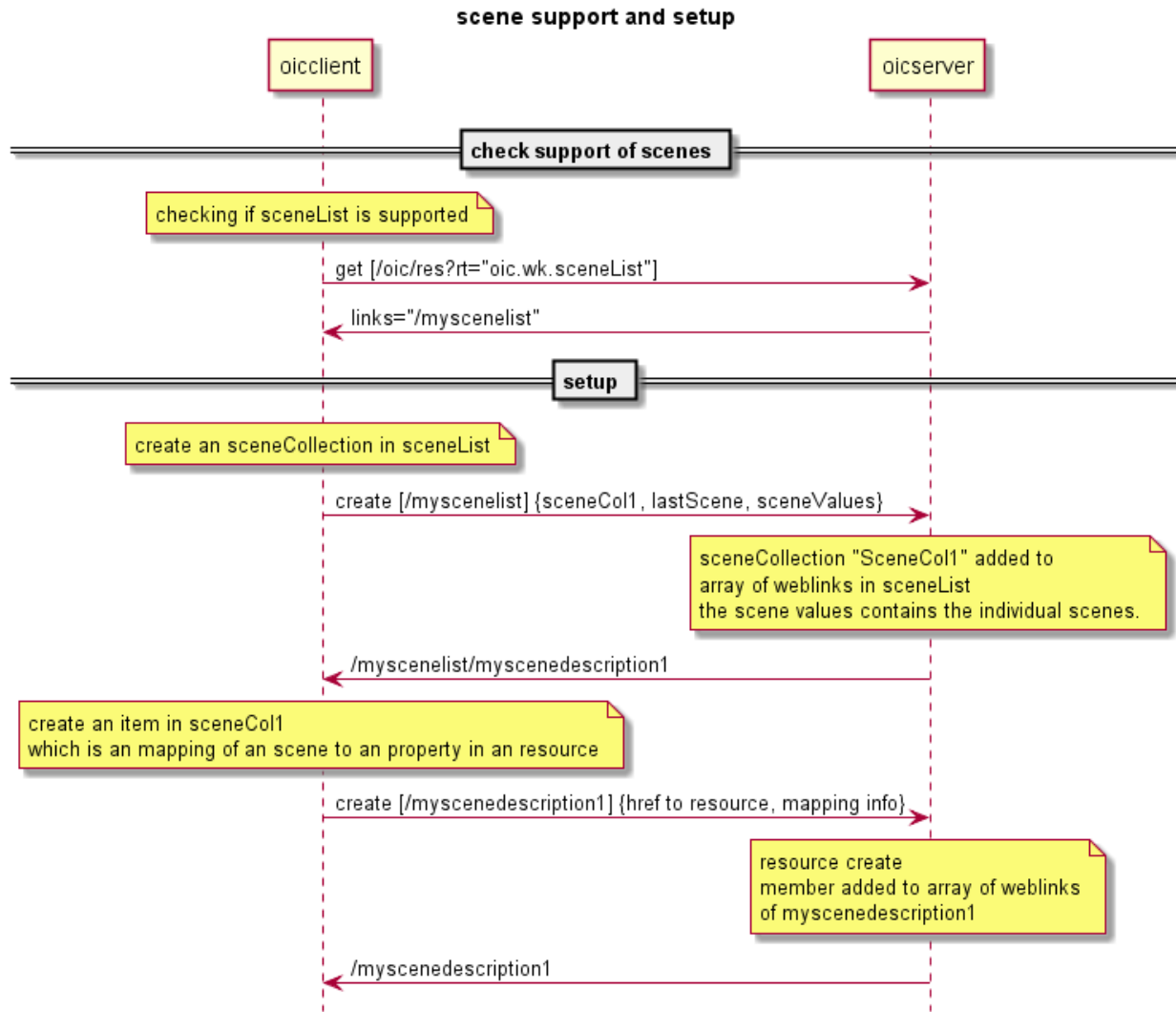


2482

2483 **Figure 33 Generic scene resource structure**

2484 **11.6.2.2 Scene creation**

2485 A Client desiring to interact with scenes needs to first determine if the server supports the scene  
2486 feature; the sceneMembers of a scene do not have to be co-located on the server supporting the  
2487 scene feature. This can be done by checking if /oic/res contains the rt of the sceneList resource.  
2488 This is depicted in first steps of Figure 34. The sceneCollection is created by the Server using  
2489 some out of bound mechanism, Client creation of scenes is not supported at this time. This will  
2490 entail defining the scene with an applicable list of scene values and the mappings for each  
2491 Resource being part of the scene. The mapping for each resource being part of the sceneCollection  
2492 is described by a resource called sceneMember. The sceneMember resource contains the link to  
2493 a resource and the mapping between the scene listed in the sceneValues property and the actual  
2494 resource property value of the Resource indicated by the link.



2495

2496

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

2497

### 11.6.2.3 Interacting with Scenes

2498

2499

2500

2501

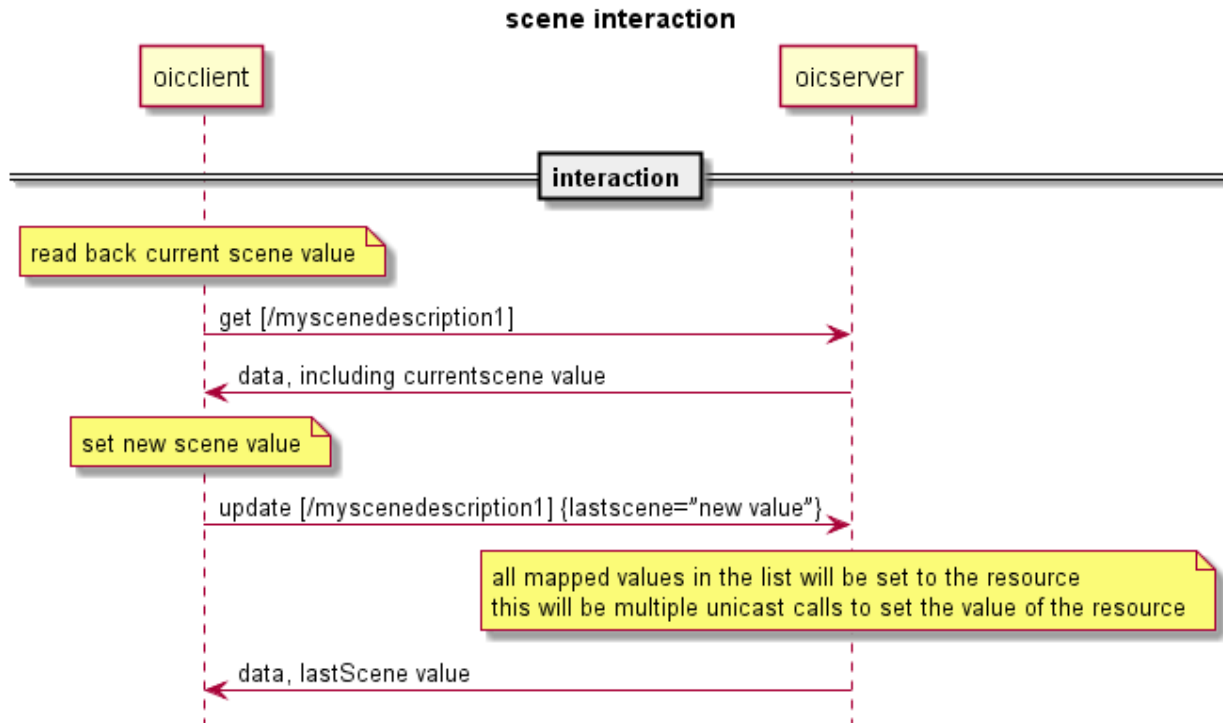
2502

2503

2504

2505

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

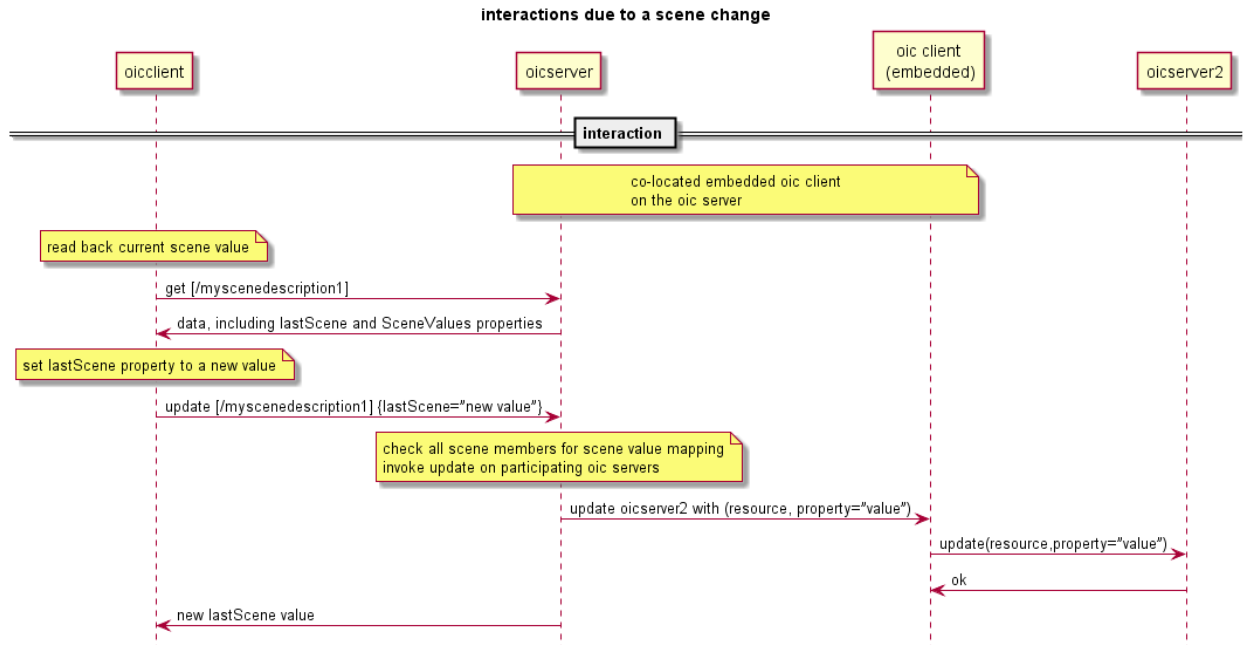


2506

2507

**Figure 35 Client interactions on a specific scene**

2508 As described previously, a scene can reference one or more resources that are present on one or  
 2509 more Servers. The scene members are re-evaluated each time a scene change takes place. This  
 2510 evaluation is triggered by a Client that is either embedded as part of the Server hosting the scene,  
 2511 or separate to the server having knowledge of the scene via a RETRIEVE operation, observing the  
 2512 referenced resources using the mechanism described in section 11.4.2. During the evaluation the  
 2513 mappings for the new scene value will be applied to the Server. This behaviour is depicted in  
 2514 Figure 36.



2515

2516

**Figure 36 Interaction overview due to a Scene change**

2517

**11.6.2.4 Summary of resource types defined for Scene functionality**

2518

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

2519

**Table 22 list of resource types for Scenes**

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

2520

**11.6.3 Security considerations**

2521

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

2526

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

2527

2528

2529



## 2530 12 Messaging

### 2531 12.1 Introduction

2532 This section specifies the protocol messaging mapping to the CRUDN messaging operations  
2533 (Section 8) for each messaging protocol specified (e.g., CoAP.). Mapping to additional protocols  
2534 is expected in later version of this specification. All the property information from the resource  
2535 model shall be carried within the message payload. This payload shall be generated in the resource  
2536 model layer and shall be encapsulated in the data connectivity layer. The message header shall  
2537 only be used to describe the message payload (e.g., verb, mime-type, message payload format),  
2538 in addition to the mandatory header fields defined in messaging protocol (e.g., CoAP) specification.  
2539 If the message header does not support this, then this information shall also be carried in the  
2540 message payload. Resource model information shall not be included in the message header  
2541 structure unless the message header field is mandatory in the messaging protocol specification.

### 2542 12.2 Mapping of CRUDN to CoAP

#### 2543 12.2.1 Overview

2544 A Device implementing CoAP shall conform to IETF RFC 7252 for the methods specified in section  
2545 12.2.3. A Device implementing CoAP shall conform to IETF draft-ietf-core-observe-16 to  
2546 implement the CoAP Observe option. Support for CoAP block transfer when the payload is larger  
2547 than the MTU is defined in section 12.2.6.

#### 2548 12.2.2 URIs

2549 An OCF: URI is mapped to a coap: URI by replacing the scheme name 'oic' with 'coap' if unsecure  
2550 or 'coaps' if secure before sending over the network by the requestor. Similarly on the receiver  
2551 side, the scheme name is replaced with 'oic'.

#### 2552 12.2.3 CoAP method with request and response

##### 2553 12.2.3.1 Overview

2554 Every request has a CoAP method that realizes the request. The primary methods and their  
2555 meanings are shown in Table 23, which provides the mapping of GET/PUT/POST/DELETE  
2556 methods to CREATE, RETRIEVE, UPDATE, and DELETE operations. The associated text provides  
2557 the generic behaviours when using these methods, however resource interfaces may modify these  
2558 generic semantics.  
2559

2560 **Table 23. CoAP request and response**

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

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

2561

2562 **12.2.3.2 CREATE with POST or PUT**

2563 **12.2.3.2.1 With POST**

2564 POST shall be used only in situations where the request URI is valid, that is it is the URI of an  
 2565 existing Resource on the Server that is processing the request. If no such Resource is present,  
 2566 the Server shall respond with an error response code of 4.xx. The use of POST for CREATE shall  
 2567 use an existing request URI which identifies the Resource on the Server responsible for creation.  
 2568 The URI of the created Resource is determined by the Server and provided to the Client in the  
 2569 response.

2570 A Client shall include the representation of the new Resource in the request payload. The new  
 2571 resource representation in the payload shall have all the necessary properties to create a valid  
 2572 Resource instance, i.e. the created Resource should be able to properly respond to the valid  
 2573 Request with mandatory Interface (e.g., GET with ?if=oic.if.baseline).

2574 Upon receiving the POST request, the Server shall either

- 2575 • create the new Resource with a new URI, respond with the new URI for the newly created  
 2576 Resource and a success response code (2.xx); or
- 2577 • respond with an error response code (4.xx).

2578 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or  
 2579 guaranteed.

2580 **12.2.3.2.2 With PUT**

2581 PUT shall be used to create a new Resource or completely replace the entire representation of an  
 2582 existing Resource. The resource representation in the payload of the PUT request shall be the  
 2583 complete representation. PUT for CREATE shall use a new request URI identifying the new  
 2584 Resource to be created.

2585 The new resource representation in the payload shall have all the necessary properties to create  
 2586 a valid Resource instance, i.e. the created Resource should be able to properly respond to the  
 2587 valid Request with mandatory Interface (e.g. GET with ?if=oic.if.baseline).

2588 Upon receiving the PUT request, the Server shall either

- 2589 • create the new Resource with the request URI provided in the PUT request and send back a  
 2590 response with a success response code (2.xx); or
- 2591 • respond with an error response code (4.xx).

2592 PUT is an unsafe method but it is idempotent, thus when a PUT request is repeated the outcome  
 2593 is the same each time.

2594 **12.2.3.3 RETRIEVE with GET**

2595 GET shall be used for the RETRIEVE operation. The GET method retrieves the representation of  
 2596 the target Resource identified by the request URI.

2597 Upon receiving the GET request, the Server shall either

2598 • send back the response with the representation of the target Resource with a success response  
2599 code (2.xx); or

2600 • respond with an error response code (4.xx) or ignore it (e.g. non-applicable multicast GET).

2601 GET is a safe method and is idempotent.

2602 **12.2.3.4 UPDATE with POST**

2603 POST shall be used only in situations where the request URI is valid, that is it is the URI of an  
2604 existing Resource on the Server that is processing the request. If no such Resource is present,  
2605 the Server shall respond with an error response code of 4.xx. A client shall use POST to UPDATE  
2606 Property values of an existing Resource (see Sections 3.1.32 and 8.4.2).

2607 Upon receiving the request, the Server shall either

- 2608 • apply the request to the Resource identified by the request URI in accordance with the applied  
2609 interface (i.e. POST for non-existent Properties is ignored) and send back a response with a  
2610 success response code (2.xx); or
- 2611 • respond with an error response code (4.xx). Note that If the representation in the payload is  
2612 incompatible with the target Resource for POST using the applied interface (i.e. the "overwrite"  
2613 semantic cannot be honored because of read-only property in the payload), then the error  
2614 response code 4.xx shall be returned.

2615 POST is unsafe and is the supported method when idempotent behaviour cannot be expected or  
2616 guaranteed.

2617 **12.2.3.5 DELETE with DELETE**

2618 DELETE shall be used for DELETE operation. The DELETE method requests that the resource  
2619 identified by the request URI be deleted.

2620 Upon receiving the DELETE request, the Server shall either

- 2621 • delete the target Resource and send back a response with a success response code (2.xx); or
- 2622 • respond with an error response code (4.xx).

2623 DELETE is unsafe but idempotent (unless URIs are recycled for new instances).

2624  
2625

2626 **12.2.4 Content Type negotiation**

2627 The Device framework mandates support of CBOR, however it allows for negotiation of the payload  
2628 body if more than one encoding type is supported by an implementation. In this case the accept  
2629 option defined in section 5.10.4 of IETF RFC 7252 shall be used to indicate which content  
2630 encodings are requested by the Client.

2631 Content types supported are as shown in Table 24.

2632 **Table 24. Content Types and Content Formats**

| Content Type    | Content Format |
|-----------------|----------------|
| application/xml | 41             |

|  |    |
|--|----|
| <b>application/exi</b>                               | 47 |
| <b>application/json<br/>defined in IETF RFC 7159</b> | 50 |
| <b>application/cbor<br/>defined in IETF RFC 7049</b> | 60 |

2633 Note: An OCF vertical can mandate a specific content type.

2634 Server and Client shall send a Content-Format option every time in a message with a payload  
2635 body. The Content Format option shall use the Content Format numeric value from Table 24.

### 2636 **12.2.5 CRUDN to CoAP response codes**

2637 The mapping of CRUDN operations response codes to CoAP response codes are identical to the  
2638 response codes defined in IETF RFC 7252.

### 2639 **12.2.6 CoAP block transfer**

2640 Basic CoAP messages work well for the small payloads typical of light-weight, constrained IoT  
2641 devices. However scenarios can be envisioned in which an application needs to transfer larger  
2642 payloads.

2643 CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be used by all Servers  
2644 which generate a content payload that would exceed the size of a CoAP datagram as the result of  
2645 handling any defined CRUDN operation.

2646 Similarly, CoAP block-wise transfer as defined in IETF draft-ietf-core-block-18 shall be supported  
2647 by all Clients. The use of block-wise transfer is applied to both the reception of payloads as well  
2648 as transmission of payloads that would exceed the size of a CoAP datagram.

2649 All blocks that are sent using this mechanism for a single instance of a transfer shall all have the  
2650 same reliability setting (i.e. all confirmable or all non-confirmable).

2651 A Client may support both the block1 (as descriptive) and block2 (as control) options as described  
2652 by IETF draft-ietf-core-block-18. A Server may support both the block1 (as control) and block2 (as  
2653 descriptive) options as described by IETF draft-ietf-core-block-18.

### 2654 **12.2.7 CoAP serialization over TCP**

#### 2655 **12.2.7.1 Introduction**

2656 In environments where TCP is already available, CoAP can take advantage of it to provide  
2657 reliability. Also in some environments UDP traffic is blocked, so deployments may use TCP. For  
2658 example, consider a cloud application acting as a Client and the Server is located at the user's  
2659 home. The Server which already support CoAP as a messaging protocol (e.g., Smart Home vertical  
2660 profile) could easily support CoAP serialization over TCP rather than adding another messaging  
2661 protocol. A Device implementing CoAP Serialization over TCP shall conform to IETF draft-  
2662 tschofenig-core-coap-tcp-tls-04.

#### 2663 **12.2.7.2 Indication of support**

2664 If UDP is blocked, clients depend on the pre-configured details on the device to find support for  
2665 CoAP over TCP. If UDP is not-blocked, a Device which supports CoAP serialization over TCP shall  
2666 populate the Messaging Protocol (mpro) property in oic/res with the value "coap+tcp" or "coaps+tcp"  
2667 to indicate that the device supports messaging protocol as specified by section 11.3.4.

2668 **12.2.7.3 Message type and header**

2669 The message type transported between Client and Server shall be a non-confirmable message  
 2670 (NON). The protocol stack used in this scenario shall be as described in section 3 in IETF draft-  
 2671 tschofenig-core-coap-tls-04.

2672 The CoAP header as described in figure 6 in IETF draft-tschofenig-core-coap-tls-04 shall be  
 2673 used for messages transmitted between a Client and a Server. A Device shall use “Alternative L3”  
 2674 as defined in IETF draft-tschofenig-core-coap-tls-04.

2675 **12.2.7.4 URI scheme**

2676 The URI scheme used shall be as defined in section 6 in IETF draft-tschofenig-core-coap-tls-  
 2677 04].

2678 For the “coaps+tcp” URI scheme the “TLS Application Layer Protocol Negotiation Extension”  
 2679 IETF RFC 7301 shall be used.

2680 **12.2.7.5 KeepAlive**

2681 **12.2.7.5.1 Overview**

2682 In order to ensure that the connection between a Device is maintained, when using CoAP  
 2683 serialization over TCP, a Device that initiated the connection should send application layer  
 2684 KeepAlive messages. The reasons to support application layer KeepAlive are as follows:

- 2685 • TCP KeepAlive only guarantees that a connection is alive at the network layer, but not at the  
 2686 application layer
- 2687 • Interval of TCP KeepAlive is configurable only using kernel parameters, and is OS dependent  
 2688 (e.g., 2 hours by default in Linux)

2689 **12.2.7.5.2 KeepAlive Mechanism**

2690 Devices supporting CoAP over TCP shall use the following KeepAlive mechanism. A Server shall  
 2691 support a resource of type oic.wk.ping as defined in Table 25.

2692 **Table 25. Ping resource**

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

2693  
 2694 Table 26 defines oic.wk.ping resource type.

2695 **Table 26. oic.wk.ping resource type definition**

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

2696 The following steps detail the KeepAlive mechanisms for a Client and Server:

- 2697 1) A Client which wants to keep the connection with a Server alive shall send a PUT request to  
2698 /oic/ping resource on the Server updating its connection Interval.
- 2699 a) This time interval shall start from 2 minutes and increases in multiples of 2 up to a maximum  
2700 of 64 minutes. It stays at 64 minutes from that point.
- 2701 2) An Server receiving this ping request shall respond within 1 minute.
- 2702 3) If a Client does not receive the response within 1 minute, it shall terminate the connection.
- 2703 4) If an Server does not receive a PUT request to ping resource within the specified "interval"  
2704 time, the Server shall terminate the connection.

2705 An example of the KeepAlive mechanism is as follows:

- 2706 • Client → Server: PUT /oic/ping {interval: 2}
- 2707 • Server → Client: 2.03 valid

2708

### 2709 **12.3 Payload Encoding in CBOR**

2710 OCF implementations shall perform the conversion to CBOR from JSON defined schemas and to  
2711 JSON from CBOR in accordance with IETF RFC 7049 section 4 unless otherwise specified in this  
2712 section.

2713 Properties defined as a JSON integer shall be encoded in CBOR as an integer (CBOR major types  
2714 0 and 1). Properties defined as a JSON number shall be encoded as an integer, single- or double-  
2715 precision floating point (CBOR major type 7, sub-types 26 and 27); the choice is implementation  
2716 dependent. Half-precision floating point (CBOR major 7, sub-type 25) shall not be used. Integer  
2717 numbers shall be within the open range  $(-2^{53}, 2^{53})$ . Properties defined as a JSON number  
2718 should be encoded as integers whenever possible; if this is not possible Properties defined as a  
2719 JSON number should use single-precision if the loss of precision does not affect the quality of  
2720 service, otherwise the Property shall use double-precision.

2721

2722 On receipt of a CBOR payload, an implementation shall be able to interpret CBOR integer values  
2723 in any position. If a property defined as a JSON integer is received encoded other than as an  
2724 integer, the implementation may reject this encoding using a final response as appropriate for the  
2725 underlying transport (e.g. 4.00 for CoAP) and thus optimise for the integer case. If a property is  
2726 defined as a JSON number an implementation shall accept integers, single- and double-precision  
2727 floating point.

## 2728 **13 Security**

2729 The details for handling security and privacy are specified in [OCF Security].

2730

## 2731 **14 Multi resource model support**

### 2732 **14.1 Interoperability issue**

#### 2733 **14.1.1 Multiple IoT Standards**

2734 Note: Alignment and interoperability between models will be added in a later version of the  
2735 specification.

2736 IoT requires standardization for interoperability among diverse devices and multiple standards are  
2737 under development currently. IETF defines network and web transfer protocol (e.g. 6lowpan  
2738 [RFC6775] and CoAP [RFC6690], [RFC7252]), oneM2M [oneM2M] produces technical

2739 specifications for a common M2M Service Layer [oneM2M-TS0001], [oneM2M-TS0004] and IPSO  
2740 Alliance [IPSO] publishes Smart Object Guideline [IPSOSmartObjects].

2741 Multitude of IoT standards are based on "Representational State Transfer (REST)", which is a  
2742 software architecture style with a coordinated set of constraints for the design of components in a  
2743 distributed hypermedia system [REST]. In REST based IoT, a real world entity is represented as  
2744 resource in a server, which a client accesses and manipulates the resource through  
2745 representations to interact with the entity, i.e. sensing and controlling the physical environments.  
2746 Moreover several IoT standards adopt the common network and web transfer protocols. oneM2M,  
2747 IPSO and OCF all use CoAP and IP/ UDP, [oneM2M-TS0008], [IPSO], [OCF] so any client and  
2748 server supporting those standards can exchange request and response messages.

2749 However in order to interact properly, it's not sufficient for IoT devices to be able to transfer CoAP  
2750 messages. IoT devices should understand each other's resources and be aware of their semantic  
2751 meaning and syntactic form. Currently each standard defines its own "resource model" and  
2752 specifies a different scheme to construct resources from physical entities such as light [OCF],  
2753 [IPSOFramework], [IPSOSmartObjects], [oneM2M-TS0001]. Hence client and server adopting  
2754 different standards can't perform meaningful interaction, i.e. the client can't manipulate the  
2755 resource representation in the server.

2756 For wider interoperability among multiple standards, IoT devices need to understand each other's  
2757 resource model to process CoAP request and response message properly. To interpret resources  
2758 correctly, client and server need to determine which resource model each other follows in the first  
2759 place. The client should be aware of whether its corresponding server adopts oneM2M or OCF  
2760 model and vice versa.

#### 2761 **14.1.2 Different resource models**

2762 OCF specification follows a resource oriented architecture with RESTful architectural style.  
2763 Without common understanding on resource model, two IoT devices can't interact with each other.

2764 Currently multiple organizations such as OCF, IPSO Alliance or oneM2M, define their own resource  
2765 model in difference ways, which may restrict interoperability to the respective ecosystems. The  
2766 main discrepancies are as follows

- 2767 • **Resource structure:** Some define resource to have attributes (e.g. oneM2M), whereas  
2768 others define it atomic and not decomposed into attributes (e.g. IPSO alliance). For  
2769 example, a smart light may be represented as a resource with on-off attribute or a  
2770 resourcecollection with on-off resource. In the former, on-off attribute doesn't have URI  
2771 and should be accessed indirectly via the resource. In the latter, being a resource itself,  
2772 on-off resource is assigned its own URI and can be directly manipulated.
- 2773 • **Resource name & type:** Some allow resource to be named freely and indicate its  
2774 characteristic with separate resource type attribute (e.g. oneM2M). Whereas others fix the  
2775 name of resource a priori and indicate its characteristic with the name itself (e.g.  
2776 IPSOalliance). For example, smart light can be named anyway such as 'LivingRoomLight\_1'  
2777 in oneM2M but should have the fixed Object name with numerical Object ID of "IPSO Light  
2778 Control (3311)" in IPSO alliance. Furthermore, in consequence, it's likely that data path in  
2779 URI is freely defined in the former and predetermined for the latter.
- 2780 • **Resource hierarchy:** Some allow resource to be organized in hierarchy so that resource  
2781 includes another resource in itself with parent-child relationship (e.g. oneM2M). Whereas  
2782 others mandate resource to be of flat structure and associate with other resources only by  
2783 referencing their links.

2784 In addition to the above, different organizations use different syntax and have different features  
2785 (e.g. resource interface), which will inhibit IoT interoperability. When IoT client and server don't  
2786 understand the resource model each supports, they can't perform RESTful transaction.

2787 For example, a smart light can be represented as an IPSO Smart Object in JSON as below:

2788

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

2789

2790

2791 In the above, "3311" is an "Object ID" defining object type, 0 an "Object Instance", designating  
2792 one or more instances, "5850", "5851", "Resource ID", defining resource type. Also IPSO embeds  
2793 resource information in data path, so "On/Off" resource has predetermined data path of  
2794 "3311/0/5850" and "Dimmer" resource datapath of "3311/0/5851"

2795

2796 Whereas the same smart light may be represented in OCF as two Resources.

2797

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

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

2798

2799



2800 **14.2 A scheme to exchange resource model information**

2801 **14.2.1 A scheme to exchange resource model information**

2802 IoT devices, i.e. client and server, need to understand the resource model which their  
2803 corresponding device supports to be able to interoperate each other.

2804 For the initial step, it would help for IoT devices to indicate resource model each device supports.  
2805 Then client and server may choose a common resource model for interaction, or in the absence of  
2806 such a common model, rely on translation between the models, possibly with the assistance of 3rd  
2807 party such as intermediary. Alignment and interoperability between models will be added in a later  
2808 version of the specification.

2809 This document presents a scheme for CoAP endpoints, client and server, to exchange resource  
2810 model they support.

2811 First, the Internet media type and Content-Format identifier are used to indicate a specific resource  
2812 model. The Internet media types can be defined to indicate the resource models, potentially with  
2813 content-coding, such as "application/ips+json", then assigned numeric Content-Format identifiers  
2814 such as "123123" to minimize payload overhead for CoAP usage.

2815 Second, CoAP Accept and Content-Format Option are used to exchange the Content-Format  
2816 identifiers indicating the resource models which CoAP endpoints prefer or support. A client  
2817 includes the CoAP Accept option to inform a server which resource model, potentially with content-  
2818 encoding, is acceptable and the server returns the payload in the preferred resource model if  
2819 available. The Content-Format Option indicates the resource model which the payload follows.

2820

2821  
2822  
2823  
2824

## Annex A (informative)

### Operation Examples

#### 2825 A.1 Introduction

2826 This section describes some example scenarios using sequence of operations between the entities  
2827 involved. In all the examples below “Light” is a Server and “Smartphone” is a Client. In one of the  
2828 scenario “Garage” additionally acts as a Server. All the examples are based on the following  
2829 example resource definitions:

2830 `rt=oc.example.light` with resource type definition as illustration in Table 27.

2831 **Table 27. `oc.example.light` resource type definition**

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

2832

2833 `rt=oc.example.garagedoor` with resource type definition as illustration in Table 28.

2834 **Table 28. `oc.example.garagedoor` resource type definition**

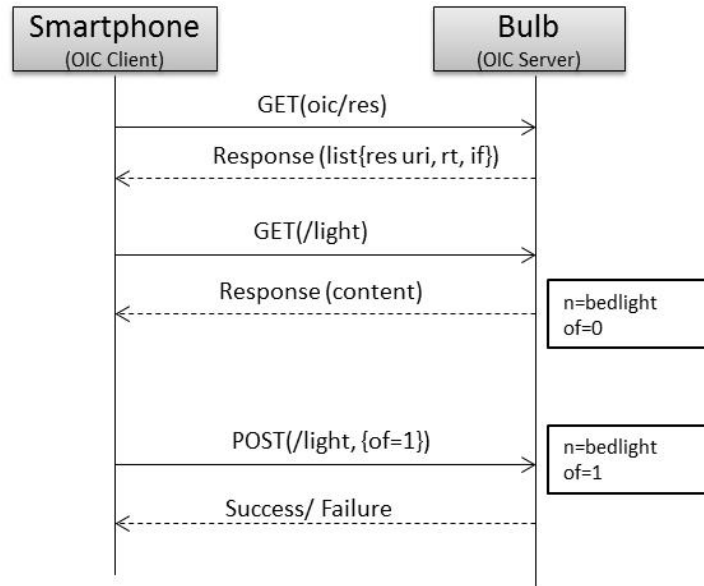
| Property title    | Property name | Value type | Value rule | Unit | Access mode | Mandatory | Description                                  |
|-------------------|---------------|------------|------------|------|-------------|-----------|--|
| <b>Name</b>       | n             | string     |      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

2835

2836 `/oic/mnt (rt=oc.wk.mnt)` used in below examples is defined in section 11.5.1.

#### 2837 A.2 When at home: From smartphone turn on a single light

2838 This sequence highlights (Figure 37) the discovery and control of an OCF light resource from an  
2839 OCF smartphone.



2840

2841

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

2842 Discovery request can be sent to “All OCF Nodes” Multicast address FF0X::158 or can be sent  
2843 directly to the IP address of device hosting the light resource.

- 2844 1) Smartphone sends a GET request to /oic/res resource to discover all resources hosted on  
2845 targeted end point
- 2846 2) The end point (bulb) responds with the list of resource URI, resource type and interfaces  
2847 supported on the end point (one of the resource is '/light' whose rt=oc.example.light)
- 2848 3) Smartphone sends a GET request to '/light' resource to know its current state
- 2849 4) The end point responds with representation of light resource ({n=bedlight;of=0})
- 2850 5) Smartphone changes the 'of' property of the light resource by sending a POST request to '/light'  
2851 resource ({of=1})
- 2852 6) On Successful execution of the request, the end point responds with the changed resource  
2853 representation. Else, error code is returned. Details of the error codes are defined in section  
2854 12.2.5.

2855 **A.3 GroupAction execution**

2856 This example will be added when groups feature is added in later version of specification

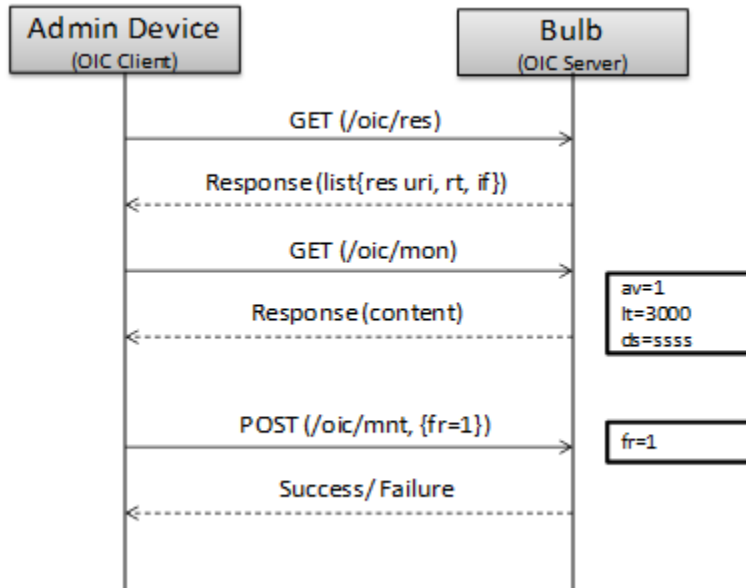
2857 **A.4 When garage door opens, turn on lights in hall; also notify smartphone**

2858 This example will be added when scripts feature is added in later version of specification

2859 **A.5 Device management**

2860 This sequence highlights (Figure 38) the device management function of maintenance.

2861



2862

2863

**Figure 38. Device management (maintenance)**

2864 **Pre-Condition:** Admin device has different security permissions and hence can perform device  
2865 management operations on the Device

- 2866 1) Admin device sends a GET request to /oic/res resource to discover all resources hosted on a  
2867 targeted end point (in this case Bulb)
- 2868 2) The end point (bulb) responds with the list of resource URI, resource type and interfaces  
2869 supported on the end point (one of the resources is /oic/mnt whose rt=oc.wk.mnt)
- 2870 3) Admin Device changes the 'fr' property of the maintenance resource by sending a POST  
2871 request to /oic/mnt resource ({fr=1}). This triggers a factory reset of the end point (bulb)
- 2872 4) On successful execution of the request, the end point responds with the changed resource  
2873 representation. Else, error code is returned. Details of the error codes are defined in section  
2874 12.2.5.

2875  
2876  
2877  
2878

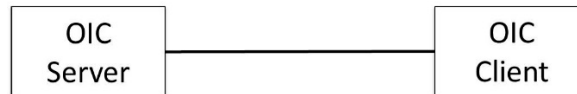
## Annex B (informative)

### OCF interaction scenarios and deployment models

#### 2879 B.1 OCF interaction scenarios

2880 A Client connects to one or multiple Servers in order to access the resources provided by those  
2881 Servers. The following are scenarios representing possible interactions among Roles:

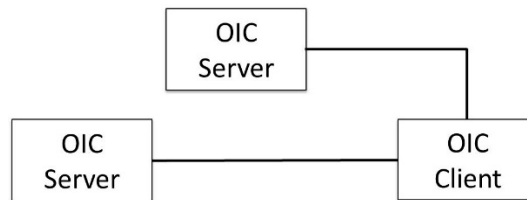
- 2882 • Direct interaction between Client and Server (Figure 39). In this scenario the Client and the  
2883 Server directly communicate without involvement of any other Device. A smartphone which  
2884 controls an actuator directly uses this scenario.



2885

2886 **Figure 39. Direct interaction between Server and Client**

- 2887 • Interaction between Client and Server using another server (Figure 40). In this scenario,  
2888 another Server provides the support needed for the Client to directly access the desired  
2889 resource on a specific Server. This scenario is used for example, when a smartphone first  
2890 accesses a discovery server to find the addressing information of a specific appliance, and  
2891 then directly accesses the appliance to control it.



2892

2893 **Figure 40. Interaction between Client and Server using another Server**

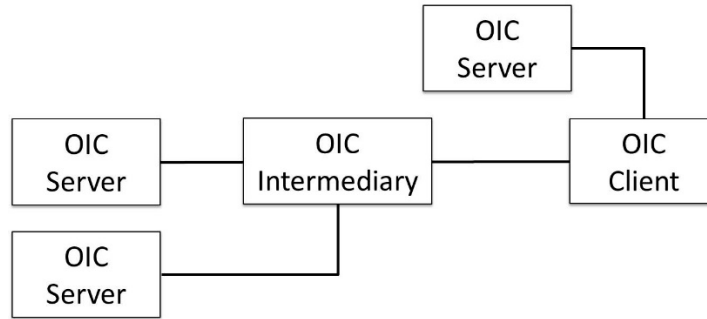
- 2894 • Interaction between Client and Server using Intermediary (Figure 41). In this scenario an  
2895 Intermediary facilitates the interaction between the Client and the Server. A smartphone which  
2896 controls appliances in a smart home via MQTT broker uses this scenario.



2897

2898 **Figure 41. Interaction between Client and Server using Intermediary**

- 2899 • Interaction between Client and Server using support from multiple Servers and intermediary  
2900 (Figure 42). In this scenario, both Server and Intermediary roles are present to facilitate the  
2901 transaction between the Client and a specific Server. An example scenario is when a  
2902 smartphone first accesses a Resource Directory (RD) server to find the address to a specific  
2903 appliance, then utilizes MQTT broker to deliver a command message to the appliance. The  
2904 smartphone can utilize the mechanisms defined in CoRE Resource Directory such as default  
2905 location, anycast address or DHCP (IETF draft-ietf-core-resource-directory-02) to discover the  
2906 Resource Directory information.

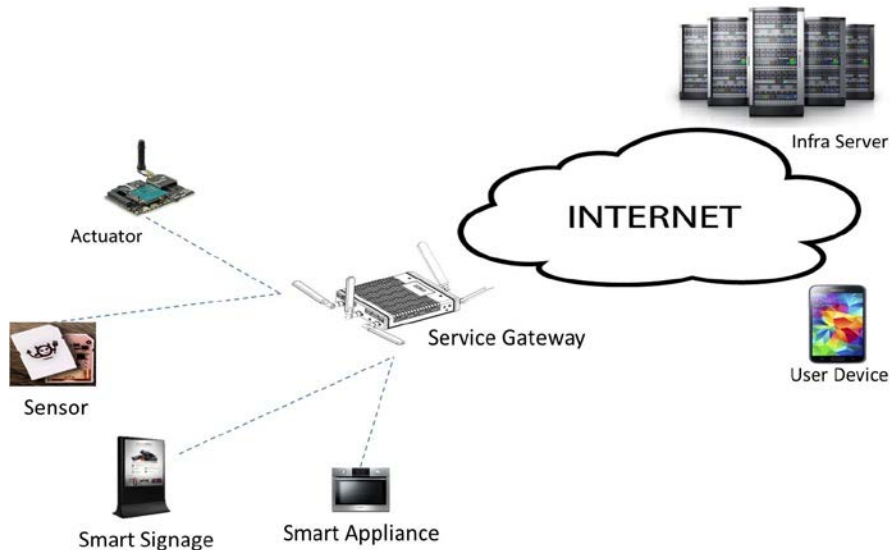


2907

2908 **Figure 42. Interaction between Client and Server using support from multiple Servers and**  
 2909 **Intermediary**

2910 **B.2 Deployment model**

2911 In deployment, Devices are deployed and interact via either wired or wireless connections. Devices  
 2912 are the physical entities that may host resources and play one or more Roles. There is no constraint  
 2913 on the structure of a deployment or number of Devices in it. Architecture is flexible and scalable  
 2914 and capable of addressing large number of devices with different device capabilities, including  
 2915 constrained devices which have limited memory and capabilities. Constrained devices are defined  
 2916 and categorized in [TCNN].



2917

2918 **Figure 43. Example of Devices**

2919 Figure 43 depicts a typical deployment and set of Devices, which may be divided in the following  
 2920 categories:

- 2921 • **Things:** Networked devices which are able to interface with physical environments. Things are  
 2922 the devices which are primarily controlled and monitored. Examples include smart appliances,  
 2923 sensors, and actuators. Things mostly take the role of Server but they may also take the role of  
 2924 Client, for example in machine-to-machine communications.
- 2925 • **User Devices:** Devices employed by the users enabling the users to access resources and  
 2926 services. Examples include smart phones, tablets, and wearable devices. User Devices mainly  
 2927 take the role of Client, but may also take the role of Server or Intermediary.

- 2928 • **Service Gateways:** Network equipment which take the role of Intermediary. Examples are  
2929 home gateways.
- 2930 • **Infra Servers:** Data centers residing in cloud infrastructure, which facilitate the interaction  
2931 among Devices by providing network services such as AAA, NAT traversal or discovery. It can  
2932 also play the role of Client or Intermediary

## Annex C (informative)

### Other Resource Models and OCF Mapping

#### C.1 Multiple resource models

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

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

- **Resource structure:** Resources may be defined to have properties (e.g., oneM2M defined resources), or may be defined as an atomic entity and not be decomposable into properties (e.g., IPSO alliance defined resources). For example, a smart light may be represented as a resource with an on-off property or a resource collection containing an on-off resource. In the former, on-off property doesn't have a URI of its own and can only be accessed indirectly via the resource. In the latter, being a resource itself, on-off resource is assigned its own URI and can be directly manipulated.
- **Resource name & type:** Resources may be allowed to be named freely and have their characteristics indicated using a resource type property (e.g., as defined in oneM2M). Alternatively, the name of resources may be defined a priori in a way that the name by itself is indicative of its characteristic (e.g., as defined by IPSO alliance). For example, in oneM2M resource model, a smart light can be named with no restrictions, such as 'LivingRoomLight\_1' but in IPSO alliance resource model it is required to have the fixed Object name with numerical Object ID of "IPSO Light Control (3311)". Consequently, it's likely that in the former case the data path in URI is freely defined and in the latter case it is predetermined.
- **Resource hierarchy:** Resources may be allowed to be organized in hierarchy where a resource contains another resource with a parent-child relationship (e.g., in oneM2M definition of resource model). Resources may also be required to have a flat structure and associate with other resources only by referencing their links.

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

#### C.2 OCF approach for support of multiple resource models

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

The following OCF characteristics enable support of other resource models:

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



2979 **C.3 Resource model indication**

2980 The Client and server exchange the information about what resource model(s) each supports.  
2981 Based on the exchanged information, the Client and Server choose a resource model to perform  
2982 RESTful interactions or to perform translation. The exchange could be part of discovery and  
2983 negotiation. Based on the exchange, the Client and Server follow a procedure to ensure  
2984 interoperability among them. They may choose a common resource model or execute translation  
2985 between resource models.

- 2986 • **Resource model schema exchange:** The Client and Server may share the resource model  
2987 information when they initiate a RESTful interaction. They may exchange the information about  
2988 which resource model they support as part of session establishment procedures. Alternatively,  
2989 each request or response message may carry the indication of which resource model it is using.  
2990 For example, [COAP] defines “Content-Format option” to indicate the “representation format”  
2991 such as “application/json”. It’s possible to extend the Content-Format Option to indicate the  
2992 resource model used with the representation format such as “application/ipsso-json”.
- 2993 • **Ensuing procedures:** After the Client and Server exchange the resource model information,  
2994 they perform a suitable procedure to ensure interoperability among them. The simplest way is  
2995 to choose a resource model supported by both the Client and Server. In case there is no  
2996 common resource model, the Client and Server may interact through a 3rd party.

2997 In addition to translation which can be resource intensive, a method based on profiles can be used  
2998 in which an OCF implementation can accommodate multiple profiles and hence multiple  
2999 ecosystems.

- 3000 • **Resource Model Profile:** the Framework defines resource model profiles and implementers or  
3001 users choose the active profile. The chosen profile constraints the Device to strict rules in how  
3002 resources are defined, instantiated and interacted with. This would allow for interoperability with  
3003 devices from the ecosystem identified by the profile (e.g., IPSO, OneM2M etc.). Although this  
3004 enables a Device to participate in and be part of any given ecosystem, this scheme does not  
3005 allow for generic interoperability at runtime. While this approach may be suitable for resource  
3006 constrained devices, more resource capable devices are expected to support more than one  
3007 profile.

3008 **C.4 An Example Profile (IPSO profile)**

3009 IPSO defines smart objects that have specific resources and they take values determined by the  
3010 data type of that resource. The smart object specification defines a category of such objects. Each  
3011 resource represents a characteristic of the smart object being modelled.

3012 While the terms may be different, there are equivalent concepts in OCF to represent these terms.  
3013 This section provides the equivalent OCF terms and then frames the IPSO smart object in OCF  
3014 terms.

3015 The IPSO object Light Control defined in Section 16 of the IPSO Smart Objects 1.0 is used as the  
3016 reference example.

3017 **C.4.1 Conceptual equivalence**

3018 The IPSO smart object definition is equivalent to an Resource Type definition which defines the  
3019 relevant characteristics of an entity being modelled. The specific IPSO Resource is equivalent to  
3020 a Property that like an IPSO Resource has a defined data type, enumeration of acceptable values,  
3021 units, a general description and access modes (based on the Interface).

3022 The general method for developing the equivalent Resource Type from an IPSO Smart Object  
3023 definition is to ignore the Object ID and replace the Object URN with an OCF ‘.’ (dot) separated  
3024 name that incorporates the IPSO object. Alternatively the Object URN can be used as the Resource

3025 Type ID as is (as long as the URN does not contain any '.' (dots)) – using the same Object URN  
 3026 as the Resource Type ID allows for compatibility when interacting with an IPSO compliant device.  
 3027 The object URN based naming does not have any bearing for OCF to OCF interoperability and so  
 3028 the OCF format is preferred – for OCF to OCF interoperability only the data model consistency is  
 3029 required.

3030 Two models are available to render IPSO objects into OCF.

- 3031 1) One is where the IPSO Smart Object represents a Resource. In this case, the IP Smart Object  
 3032 is regarded as a resource with the Resource Type matching the description of the Smart Object.  
 3033 Furthermore, each resource in the IPSO definition is represented as an Property in the  
 3034 Resource Type (the IPSO Resource ID is replaced with a string representing the Property).  
 3035 This is the preferred approach when the IPSO Data Model is expressed in the Resource Model.
- 3036 2) The other approach is to model an IPSO Smart Object as an Collection. Each IPSO Resource  
 3037 is then modelled as an Resource with an Resource Type that matches the definition of the  
 3038 IPSO Resource. Each of these resource instances are then bound to the Collection that  
 3039 represents this IPSO Smart Object.

3040

3041 Below is an example showing how an IPSO LightControl Object is modelled as a Resource.

3042 **Resource Type: Light Control**

3043 Description: This Object is used to control a light source, such as a LED or other light. It allows a  
 3044 light to be turned on or off and its dimmer setting to be controlled as a percentage value between  
 3045 0 and 100. An optional colour setting enables a string to be used to indicate the desired colour.  
 3046 Table 29 and Table 30 define the resource type and its properties, respectively.

3047

**Table 29. Light control resource type definition**

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

3048

3049

**Table 30. Light control resource type definition**

| Property title | Property name | Value type | Value rule | Unit                              | Access mode | Mandatory | Description  |
|----------------|---------------|------------|------------|-----------------------------------|-------------|-----------|--|
| <b>On/Off</b>  | "on-off"      | boolean    |            |                                   | R, W        | yes       | On/Of Control:<br>0 = Off<br>1 = On  |
| <b>Dimmer</b>  | "dim"         | integer    |            | %                                 | R, W        | no        | Proportional Control,<br>integer value between 0<br>and 100 as percentage  |
| <b>Color</b>   | "color"       | string     | 0 – 100    | Defined by<br>"units"<br>property | R, W        | no        | String representing some<br>value in color space                           |
| <b>Units</b>   | "units"       | string     |            |                                   | R           | no        | Measurement Units<br>Definition e.g., "Cel" for<br>Temperature in Celsius. |
| <b>On Time</b> | "ontime"      | integer    |            | s                                 | R, W        | no        | The time in seconds that<br>the light has been on.                         |

|                                |           |       |  |    |   |    |  |
|--------------------------------|-----------|-------|--|----|---|----|--|
|                                |           |       |  |    |   |    | Writing a value of 0 resets the counter  |
| <b>Cumulative active power</b> | "cumap"   | float |  | Wh | R | no | The cumulative active power since the last cumulative energy reset or device start |
| <b>Power Factor</b>            | "powfact" | float |  |    | R | no | The power factor of the load   |

3050

3051

3052  
3053  
3054  
3055

## Annex D (normative)

### Resource Type definitions

#### 3056 D.1 List of resource type definitions

3057 Table 31 contains the list of defined core resources in this specification.

3058 **Table 31. Alphabetized list of core resources**

| Friendly Name (informative) | Resource Type (rt)     | Section  |
|-----------------------------|------------------------|----------|
| Collections                 | oic.wk.col             | D.2      |
| Configuration               | oic.wk.con             | D.3      |
| Device                      | oic.wk.d               | D.4      |
| Discoverable Resources      | oic.wk.res             | D.8, D.9 |
| Maintenance                 | oic.wk.mnt             | D.5      |
| Platform                    | oic.wk.p               | D.6      |
| Ping                        | oic.wk.ping            | D.7      |
| Resource Directory          | oic.wk.rd              | D.13     |
| Scenes (Top Level)          | oic.wk.sceneList       | D.10     |
| Scenes Collections          | oic.wk.sceneCollection | D.11     |
| Scenes Member               | oic.wk.sceneMember     | D.12     |

3059

#### 3060 D.2 OCF Collection

##### 3061 D.2.1 Introduction

3062 OCF Collection Resource Type contains properties and links. The oic.if.baseline interface exposes  
3063 a representation of the links and the properties of the collection resource itself

##### 3064 D.2.2 Example URI

3065 /CollectionBaselineInterfaceURI

##### 3066 D.2.3 Resource Type

3067 The resource type (rt) is defined as: oic.wk.col.

```

3068 D.2.4 RAML Definition
3069 #%RAML 0.8
3070 title: Collections
3071 version: 1.0
3072 traits:
3073   - interface-ll :
3074     queryParameters:
3075       if:
3076         enum: ["oic.if.ll"]
3077   - interface-b :
3078     queryParameters:
3079       if:
3080         enum: ["oic.if.b"]
3081   - interface-baseline :
3082     queryParameters:
3083       if:
3084         enum: ["oic.if.baseline"]
3085
3086 /CollectionBaselineInterfaceURI:
3087   description: |
3088     OCF Collection Resource Type contains properties and links.
3089     The oic.if.baseline interface exposes a representation of
3090     the links and the properties of the collection resource itself
3091
3092   is : ['interface-baseline']
3093   get:
3094     description: |
3095       Retrieve on Baseline Interface
3096
3097   responses :
3098     200:
3099       body:
3100         application/json:
3101           schema: /
3102             {
3103               "$schema": "http://json-schema.org/draft-04/schema#",
3104               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3105 reserved.",
3106               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3107 schema.json#",
3108               "title": "Collection",
3109               "definitions": {
3110                 "oic.collection.setoflinks": {
3111                   "description": "A set (array) of simple or individual OIC Links. In
3112 addition to properties required for an OIC Link, the identifier for that link in this set is also
3113 required",
3114                   "type": "array",
3115                   "items": {
3116                     "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3117                   }
3118                 },
3119                 "oic.collection.alllinks": {
3120                   "description": "All forms of links in a collection",
3121                   "oneOf": [
3122                     {
3123                       "$ref": "#/definitions/oic.collection.setoflinks"
3124                     }
3125                   ]
3126                 },
3127               "oic.collection": {

```

```

3128         "type": "object",
3129         "description": "A collection is a set (array) of tagged-link or set
3130 (array) of simple links along with additional properties to describe the collection itself",
3131         "properties": {
3132             "n": {
3133                 "type": "string",
3134                 "description": "User friendly name of the
3135 collection"
3136             },
3137             "id": {
3138                 "anyOf": [
3139                     {
3140                         "type": "integer",
3141                         "description": "A number that is unique to that
3142 collection; like an ordinal number that is not repeated"
3143                     },
3144                     {
3145                         "type": "string",
3146                         "description": "A unique string that could be a hash or
3147 similarly unique"
3148                     },
3149                     {
3150                         "$ref": "oic.types-schema.json#/definitions/uuid",
3151                         "description": "A unique string that could be a UUIDv4"
3152                     }
3153                 ],
3154                 "description": "ID for the collection. Can be a value that is
3155 unique to the use context or a UUIDv4"
3156             },
3157             "di": {
3158                 "$ref": "oic.types-schema.json#/definitions/uuid",
3159                 "description": "The device ID which is an UUIDv4 string; used for
3160 backward compatibility with Spec A definition of /oic/res"
3161             },
3162             "rts": {
3163                 "$ref": "oic.core-
3164 schema.json#/definitions/oic.core/properties/rt",
3165                 "description": "Defines the list of allowable resource types (for
3166 Target and anchors) in links included in the collection; new links being created can only be from
3167 this list"
3168             },
3169             "drel": {
3170                 "type": "string",
3171                 "description": "When specified this is the default relationship
3172 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3173             },
3174             "links": {
3175                 "$ref": "#/definitions/oic.collection.alllinks"
3176             }
3177         },
3178         "type": "object",
3179         "allOf": [
3180             {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3181             {"$ref": "#/definitions/oic.collection"}
3182         ]
3183     }
3184
3185     example: /
3186     {
3187         "rt": ["oic.wk.col"],
3188         "id": "unique_example_id",
3189         "rts": [ "oic.r.switch.binary", "oic.r.airflow" ],
3190         "links": [
3191             {
3192                 "href": "switch",
3193                 "rt": [ "oic.r.switch.binary" ],
3194                 "if": [ "oic.if.a", "oic.if.baseline" ]
3195             },
3196             {
3197                 "href": "airFlow",

```

```

3198         "rt": ["oic.r.airflow"],
3199         "if": ["oic.if.a", "oic.if.baseline"]
3200     }
3201 }
3202 }
3203
3204 post:
3205     description: |
3206         Update on Baseline Interface
3207
3208     body:
3209         application/json:
3210             schema: /
3211                 {
3212                     "$schema": "http://json-schema.org/draft-04/schema#",
3213                     "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3214 reserved.",
3215                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3216 schema.json#",
3217                     "title": "Collection",
3218                     "definitions": {
3219                         "oic.collection.setoflinks": {
3220                             "description": "A set (array) of simple or individual OIC Links. In addition
3221 to properties required for an OIC Link, the identifier for that link in this set is also required",
3222                             "type": "array",
3223                             "items": {
3224                                 "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3225                             }
3226                         },
3227                         "oic.collection.alllinks": {
3228                             "description": "All forms of links in a collection",
3229                             "oneOf": [
3230                                 {
3231                                     "$ref": "#/definitions/oic.collection.setoflinks"
3232                                 }
3233                             ]
3234                         },
3235                         "oic.collection": {
3236                             "type": "object",
3237                             "description": "A collection is a set (array) of tagged-link or set (array)
3238 of simple links along with additional properties to describe the collection itself",
3239                             "properties": {
3240                                 "n": {
3241                                     "type": "string",
3242                                     "description": "User friendly name of the
3243 collection"
3244                                 },
3245                                 "id": {
3246                                     "anyOf": [
3247                                         {
3248                                             "type": "integer",
3249                                             "description": "A number that is unique to that collection;
3250 like an ordinal number that is not repeated"
3251                                         },
3252                                         {
3253                                             "type": "string",
3254                                             "description": "A unique string that could be a hash or
3255 similarly unique"
3256                                         },
3257                                         {
3258                                             "$ref": "oic.types-schema.json#/definitions/uuid",
3259                                             "description": "A unique string that could be a UUIDv4"
3260                                         }
3261                                     ],
3262                                     "description": "ID for the collection. Can be an value that is unique
3263 to the use context or a UUIDv4"
3264                                 },
3265                                 "di": {
3266                                     "$ref": "oic.types-schema.json#/definitions/uuid",

```

```

3266         "description": "The device ID which is an UUIDv4 string; used for
3267 backward compatibility with Spec A definition of /oic/res"
3268     },
3269     "rts": {
3270         "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
3271         "description": "Defines the list of allowable resource types (for
3272 Target and anchors) in links included in the collection; new links being created can only be from
3273 this list"
3274     },
3275     "drel": {
3276         "type": "string",
3277         "description": "When specified this is the default relationship to
use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3278     },
3279     "links": {
3280         "$ref": "#/definitions/oic.collection.alllinks"
3281     }
3282 }
3283 },
3284 },
3285 "type": "object",
3286 "allOf": [
3287     {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3288     {"$ref": "#/definitions/oic.collection"}
3289 ]
3290 }
3291
3292 responses :
3293 200:
3294     body:
3295     application/json:
3296         schema: /
3297         {
3298             "$schema": "http://json-schema.org/draft-04/schema#",
3299             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3300 reserved.",
3301             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
3302 schema.json#",
3303             "title": "Collection",
3304             "definitions": {
3305                 "oic.collection.setoflinks": {
3306                     "description": "A set (array) of simple or individual OIC Links. In
3307 addition to properties required for an OIC Link, the identifier for that link in this set is also
3308 required",
3309                     "type": "array",
3310                     "items": {
3311                         "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
3312                     }
3313                 },
3314                 "oic.collection.alllinks": {
3315                     "description": "All forms of links in a collection",
3316                     "oneOf": [
3317                         {
3318                             "$ref": "#/definitions/oic.collection.setoflinks"
3319                         }
3320                     ]
3321                 },
3322                 "oic.collection": {
3323                     "type": "object",
3324                     "description": "A collection is a set (array) of tagged-link or set
3325 (array) of simple links along with additional properties to describe the collection itself",
3326                     "properties": {
3327                         "n": {
3328                             "type": "string",
3329                             "description": "User friendly name of the
3330 collection"
3331                         },
3332                         "id": {
3333                             "anyOf": [

```



```

3334         "type": "integer",
3335         "description": "A number that is unique to that
3336 collection; like an ordinal number that is not repeated"
3337     },
3338     {
3339         "type": "string",
3340         "description": "A unique string that could be a hash or
3341 similarly unique"
3342     },
3343     {
3344         "$ref": "oic.types-schema.json#/definitions/uuid",
3345         "description": "A unique string that could be a UUIDv4"
3346     }
3347 ],
3348     "description": "ID for the collection. Can be an value that is
3349 unique to the use context or a UUIDv4"
3350 },
3351     "di": {
3352         "$ref": "oic.types-schema.json#/definitions/uuid",
3353         "description": "The device ID which is an UUIDv4 string; used for
3354 backward compatibility with Spec A definition of /oic/res"
3355     },
3356     "rts": {
3357         "$ref": "oic.core-
3358 schema.json#/definitions/oic.core/properties/rt",
3359         "description": "Defines the list of allowable resource types (for
3360 Target and anchors) in links included in the collection; new links being created can only be from
3361 this list"
3362     },
3363     "drel": {
3364         "type": "string",
3365         "description": "When specified this is the default relationship
3366 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
3367     },
3368     "links": {
3369         "$ref": "#/definitions/oic.collection.alllinks"
3370     }
3371 }
3372 },
3373     "type": "object",
3374     "allOf": [
3375         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3376         {"$ref": "#/definitions/oic.collection"}
3377     ]
3378 }
3379

```

3380 **D.2.5 Property Definition**

| Property name | Value type                 | Mandatory | Access mode | Description   |
|---------------|----------------------------|-----------|-------------|---|
| rt            | array: see schema          | yes       | Read Write  | Resource Type   |
| di            | multiple types: see schema |           | Read Write  | Unique identifier for device (UUID)   |
| title         | string                     |           | Read Write  | A title for the link relation. Can be used by the UI to provide a context                                 |
| huri          | string                     |           | Read Write  | The base URI used to fully qualify a Relative Reference in the href parameter. Use the OCF Schema for URI |

|        |                               |     |            |  |
|--------|-------------------------------|-----|------------|--|
| ins    | multiple types:<br>see schema |     | Read Write | The instance identifier for this web link in an array of web links - used in collections   |
| p      | object: see<br>schema         |     | Read Write | Specifies the framework policies on the Resource referenced by the target URI  |
| href   | string                        | yes | Read Write | This is the target URI, it can be specified as a Relative Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique. |
| rel    | multiple types:<br>see schema |     | Read Write | The relation of the target URI referenced by the link to the context URI   |
| type   | array: see<br>schema          |     | Read Write | A hint at the representation of the resource referenced by the target URI. This represents the media types that are used for both accepting and emitting                     |
| anchor | string                        |     | Read Write | This is used to override the context URI e.g. override the URI of the containing collection  |
| if     | array: see<br>schema          | yes | Read Write | The interface set supported by this resource   |

3381

### D.2.6 CRUDN behavior

| Resource                        | Create | Read | Update | Delete | Notify |
|---------------------------------|--------|------|--------|--------|--------|
| /CollectionBaselineInterfaceURI |        | get  | post   |        |        |

## 3382 D.2.7 Referenced JSON schemas

### 3383 D.2.8 oic.oic-link-schema.json

```
3384 {
3385   "$schema": "http://json-schema.org/draft-04/schema#",
3386   "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
3387   "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
3388   "definitions": {
3389     "oic.oic-link": {
3390       "type": "object",
3391       "properties": {
3392         "href": {
3393           "type": "string",
3394           "maxLength": 256,
3395           "description": "This is the target URI, it can be specified as a Relative Reference or
3396 fully-qualified URI. Relative Reference should be used along with the di parameter to make it
3397 unique.",
3398           "format": "uri"
3399         },
3400         "rel": {
3401           "oneOf": [
3402             {
3403               "type": "array",
3404               "items": {
3405                 "type": "string",
3406                 "maxLength": 64
3407               },
3408               "minItems": 1,
3409               "default": ["hosts"]
3410             },
3411             {
3412               "type": "string",
3413               "maxLength": 64,
3414               "default": "hosts"
3415             }
3416           ],
3417           "description": "The relation of the target URI referenced by the link to the context URI"
3418         },
3419         "rt": {
3420           "type": "array",
3421           "items": {
3422             "type": "string",
3423             "maxLength": 64
3424           },
3425           "minItems": 1,
3426           "description": "Resource Type"
3427         },
3428         "if": {
3429           "type": "array",
3430           "items": {
3431             "type": "string",
3432             "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
3433 "oic.if.a", "oic.if.s" ]
3434           },
3435           "minItems": 1,
3436           "description": "The interface set supported by this resource"
3437         },
3438         "di": {
3439           "$ref": "oic.types-schema.json#/definitions/uuid",
3440           "description": "Unique identifier for device (UUID)"
3441         },
3442         "buri": {
3443           "type": "string",
3444           "description": "The base URI used to fully qualify a Relative Reference in the href
3445 parameter. Use the OCF Schema for URI",
3446           "maxLength": 256,
3447           "format": "uri"
3448         },
3449         "p": {
3450           "description": "Specifies the framework policies on the Resource referenced by the target
```

```

3451 URI",
3452     "type": "object",
3453     "properties": {
3454         "bm": {
3455             "description": "Specifies the framework policies on the Resource referenced by the
3456 target URI for e.g. observable and discoverable",
3457             "type": "integer"
3458         },
3459         "sec": {
3460             "description": "Specifies if security needs to be turned on when looking to interact
3461 with the Resource",
3462             "default": false,
3463             "type": "boolean"
3464         },
3465         "port": {
3466             "description": "Secure port to be used for connection",
3467             "type": "integer"
3468         }
3469     },
3470     "required" : ["bm"]
3471 },
3472     "title": {
3473         "type": "string",
3474         "maxLength": 64,
3475         "description": "A title for the link relation. Can be used by the UI to provide a
3476 context"
3477     },
3478     "anchor": {
3479         "type": "string",
3480         "maxLength": 256,
3481         "description": "This is used to override the context URI e.g. override the URI of the
3482 containing collection",
3483         "format": "uri"
3484     },
3485     "ins": {
3486         "oneOf": [
3487             {
3488                 "type": "integer",
3489                 "description": "An ordinal number that is not repeated - must be unique in the
3490 collection context"
3491             },
3492             {
3493                 "type": "string",
3494                 "maxLength": 256,
3495                 "format": "uri",
3496                 "description": "Any unique string including a URI"
3497             },
3498             {
3499                 "$ref": "oic.types-schema.json#/definitions/uuid",
3500                 "description": "Unique identifier (UUID)"
3501             }
3502         ],
3503         "description": "The instance identifier for this web link in an array of web links - used
3504 in collections"
3505     },
3506     "type": {
3507         "type": "array",
3508         "description": "A hint at the representation of the resource referenced by the target
3509 URI. This represents the media types that are used for both accepting and emitting",
3510         "items": {
3511             "type": "string",
3512             "maxLength": 64
3513         },
3514         "minItems": 1,
3515         "default": "application/cbor"
3516     }
3517 },
3518     "required": [ "href", "rt", "if" ]
3519 }
3520 },
3521     "type": "object",

```

```

3522     "allOf": [
3523     { "$ref": "#/definitions/oic.oic-link" }
3524     ]
3525 }
3526

```

## 3527 D.3 OIC Configuration

### 3528 D.3.1 Introduction

3529 Known resource that is hosted by every Server. Allows for device specific information to be  
3530 configured.

### 3531 D.3.2 Example URI

3532 /oic/con

### 3533 D.3.3 Resource Type

3534 The resource type (rt) is defined as: oic.wk.con.

### 3535 D.3.4 RAML Definition

```

3536 #%RAML 0.8
3537 title: OIC Configuration
3538 version: v1-20160622
3539 traits:
3540   - interface :
3541     queryParameters:
3542       if:
3543         enum: ["oic.if.rw", "oic.if.baseline"]
3544
3545 /oic/con:
3546   description: |
3547     Known resource that is hosted by every Server.
3548     Allows for device specific information to be configured.
3549
3550   is : ['interface']
3551   get:
3552     description: |
3553       Retrieves the current configuration settings
3554
3555   responses :
3556     200:
3557       body:
3558         application/json:
3559           schema: /
3560             {
3561               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-
3562 schema.json#",
3563               "$schema": "http://json-schema.org/draft-04/schema#",
3564               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3565 reserved.",
3566               "definitions": {
3567                 "oic.wk.con": {
3568                   "type": "object",
3569                   "properties": {
3570                     "n": {
3571                       "type": "string",
3572                       "maxLength": 64,
3573                       "description": "Human friendly name"
3574                     },
3575                     "loc": {

```

```

3576         "type": "string",
3577         "description": "Location information"
3578     },
3579     "locn": {
3580         "type": "string",
3581         "maxLength": 64,
3582         "description": "Human Friendly Name"
3583     },
3584     "c": {
3585         "type": "string",
3586         "maxLength": 64,
3587         "description": "Currency"
3588     },
3589     "r": {
3590         "type": "string",
3591         "maxLength": 64,
3592         "description": "Region"
3593     }
3594 }
3595 }
3596 },
3597 "type": "object",
3598 "allOf": [
3599     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
3600     { "$ref": "#/definitions/oic.wk.con" }
3601 ],
3602 "required": [ "n" ]
3603 }
3604

```

```

3605 example: /
3606 {
3607     "rt":  ["oic.wk.con"],
3608     "n":   "My Friendly Device Name",
3609     "loc": "My Location Information",
3610     "locn": "My Location Name",
3611     "c":   "USD",
3612     "r":   "MyRegion"
3613 }
3614

```

post:

```

3616 description: |
3617     Update the information about the Device
3618

```

```

3619 body:
3620     application/json:

```

```

3621     schema: /
3622     {
3623         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-schema.json#",
3624         "$schema": "http://json-schema.org/draft-04/schema#",
3625         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3626 reserved.",
3627         "definitions": {
3628             "oic.wk.con": {
3629                 "type": "object",
3630                 "properties": {
3631                     "n": {
3632                         "type": "string",
3633                         "maxLength": 64,
3634                         "description": "Human friendly name"
3635                     },
3636                     "loc": {
3637                         "type": "string",
3638                         "description": "Location information"
3639                     },
3640                     "locn": {
3641                         "type": "string",
3642                         "maxLength": 64,

```

```

3643         "description": "Human Friendly Name"
3644     },
3645     "c": {
3646         "type": "string",
3647         "maxLength": 64,
3648         "description": "Currency"
3649     },
3650     "r": {
3651         "type": "string",
3652         "maxLength": 64,
3653         "description": "Region"
3654     }
3655 }
3656 }
3657 },
3658 "type": "object",
3659 "allOf": [
3660     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
3661     { "$ref": "#/definitions/oic.wk.con" }
3662 ],
3663 "required": [ "n" ]
3664 }
3665
3666 example: /
3667 {
3668     "n": "My Friendly Device Name"
3669 }
3670
3671 responses :
3672 200:
3673     body:
3674         application/json:
3675             schema: /
3676                 {
3677                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.con-
3678 schema.json#",
3679                     "$schema": "http://json-schema.org/draft-04/schema#",
3680                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3681 reserved.",
3682                     "definitions": {
3683                         "oic.wk.con": {
3684                             "type": "object",
3685                             "properties": {
3686                                 "n": {
3687                                     "type": "string",
3688                                     "maxLength": 64,
3689                                     "description": "Human friendly name"
3690                                 },
3691                                 "loc": {
3692                                     "type": "string",
3693                                     "description": "Location information"
3694                                 },
3695                                 "locn": {
3696                                     "type": "string",
3697                                     "maxLength": 64,
3698                                     "description": "Human Friendly Name"
3699                                 },
3700                                 "c": {
3701                                     "type": "string",
3702                                     "maxLength": 64,
3703                                     "description": "Currency"
3704                                 },
3705                                 "r": {
3706                                     "type": "string",
3707                                     "maxLength": 64,
3708                                     "description": "Region"
3709                                 }

```

```

3710     }
3711   },
3712 },
3713 "type": "object",
3714 "allof": [
3715   { "$ref": "oic.core-schema.json#/definitions/oic.core"},
3716   { "$ref": "#/definitions/oic.wk.con" }
3717 ],
3718 "required": [ "n" ]
3719 }
3720
3721 example: /
3722 {
3723   "n": "My Friendly Device Name"
3724 }
3725

```

### 3726 D.3.5 Property Definition

| Property name | Value type | Mandatory | Access mode | Description          |
|---------------|------------|-----------|-------------|----------------------|
| loc           | string     |           | Read Write  | Location information |
| c             | string     |           | Read Write  | Currency             |
| r             | string     |           | Read Write  | Region               |
| locn          | string     |           | Read Write  | Human Friendly Name  |
| n             | string     | yes       | Read Write  | Human friendly name  |

### 3727 D.3.6 CRUDN behavior

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

## 3728 D.4 Device

### 3729 D.4.1 Introduction

3730 Known resource that is hosted by every Server. Allows for logical device specific information to be  
 3731 discovered.

### 3732 D.4.2 Wellknown URI

3733 /oic/d

### 3734 D.4.3 Resource Type

3735 The resource type (rt) is defined as: oic.wk.d.

### 3736 D.4.4 RAML Definition

```

3737 #%RAML 0.8
3738 title: OIC Root Device
3739 version: v1-20160622
3740 traits:
3741   - interface :
3742     queryParameters:
3743       if:
3744         enum: ["oic.if.r", "oic.if.baseline"]
3745
3746 /oic/d:
3747   description: |

```



```

3748     Known resource that is hosted by every Server.
3749     Allows for logical device specific information to be discovered.
3750
3751     is : ['interface']
3752
3753     get:
3754         description: |
3755             Retrieve the information about the Device
3756
3757     responses :
3758         200:
3759             body:
3760                 application/json:
3761                     schema: /
3762                         {
3763                             "$schema": "http://json-schemas.org/draft-04/schema#",
3764                             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3765 reserved.",
3766                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.d-
3767 schema.json#",
3768                             "definitions": {
3769                                 "oic.wk.d": {
3770                                     "type": "object",
3771                                     "properties": {
3772                                         "n": {
3773                                             "type": "string",
3774                                             "maxLength": 64,
3775                                             "readOnly": true,
3776                                             "description": "Human friendly name"
3777                                         },
3778                                         "di": {
3779                                             "$ref": "oic.types-schema.json#/definitions/uuid",
3780                                             "readOnly": true,
3781                                             "description": "Unique identifier for device (UUID)"
3782                                         },
3783                                         "icv": {
3784                                             "type": "string",
3785                                             "maxLength": 64,
3786                                             "readOnly": true,
3787                                             "description": "The version of the OIC Server"
3788                                         },
3789                                         "dmv": {
3790                                             "type": "string",
3791                                             "maxLength": 64,
3792                                             "readOnly": true,
3793                                             "description": "The spec version of the vertical and/or resource
3794 specification"
3795                                         }
3796                                     }
3797                                 },
3798                                 "type": "object",
3799                                 "allOf": [
3800                                     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
3801                                     { "$ref": "#/definitions/oic.wk.d" }
3802                                 ],
3803                                 "required": [ "n", "di", "icv", "dmv" ]
3804                             }
3805
3806                 example: /
3807                     {
3808                         "n": "Device 1",
3809                         "rt": ["oic.wk.d"],
3810                         "di": "54919CA5-4101-4AE4-595B-353C51AA983C",
3811                         "icv": "core.1.1.0",
3812                         "dmv": "res.1.1.0"

```

3813 }  
3814

#### 3815 D.4.5 Property Definition

| Property name | Value type                    | Mandatory | Access mode | Description  |
|---------------|-------------------------------|-----------|-------------|--|
| di            | multiple types:<br>see schema | yes       | Read Only   | Unique identifier for device (UUID)                            |
| icv           | string                        | yes       | Read Only   | The version of the OIC Server                                  |
| dmv           | string                        | yes       | Read Only   | The spec version of the vertical and/or resource specification |
| n             | string                        | yes       | Read Only   | Human friendly name  |

#### 3816 D.4.6 CRUDN behavior

| Resource | Create | Read | Update | Delete | Notify |
|----------|--------|------|--------|--------|--------|
| /oic/d   |        | get  |        |        |        |

### 3817 D.5 Maintenance

#### 3818 D.5.1 Introduction

3819 The resource through which a Device is maintained and can be used for diagnostic purposes. fr  
3820 (Factory Reset) is a boolean. The value 0 means No action (Default), the value 1 means Start  
3821 Factory Reset After factory reset, this value shall be changed back to the default value. rb (Reboot)  
3822 is a boolean. The value 0 means No action (Default), the value 1 means Start Reboot. After  
3823 Reboot, this value shall be changed back to the default value

#### 3824 D.5.2 Wellknown URI

3825 /oic/mnt

#### 3826 D.5.3 Resource Type

3827 The resource type (rt) is defined as: oic.wk.mnt.

#### 3828 D.5.4 RAML Definition

```
3829 #%RAML 0.8
3830 title: Maintenance
3831 version: v1-20160622
3832 traits:
3833   - interface-update :
3834     queryParameters:
3835       if:
3836         enum: ["oic.if.rw", "oic.if.baseline"]
3837   - interface-all :
3838     queryParameters:
3839       if:
3840         enum: ["oic.if.rw", "oic.if.r", "oic.if.baseline"]
```

```
3841 /oic/mnt:
```

```
3842   description: |
3843     The resource through which a Device is maintained and can be used for diagnostic purposes.
3844     fr (Factory Reset) is a boolean.
3845     The value 0 means No action (Default), the value 1 means Start Factory Reset
3846     After factory reset, this value shall be changed back to the default value.
3847     rb (Reboot) is a boolean.
3848     The value 0 means No action (Default), the value 1 means Start Reboot.
```

```

3850     After Reboot, this value shall be changed back to the default value
3851
3852 get:
3853     description: |
3854         Retrieve the maintenance action status
3855
3856     is : ['interface-all']
3857
3858     responses :
3859         200:
3860             body:
3861                 application/json:
3862                     schema: /
3863                         {
3864                             "$schema": "http://json-schemas.org/draft-04/schema#",
3865                             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
3866 reserved.",
3867                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
3868 schema.json#",
3869                             "definitions": {
3870                                 "oic.wk.mnt": {
3871                                     "type": "object",
3872                                     "anyOf": [
3873                                         {"required": ["fr"]},
3874                                         {"required": ["rb"]}
3875                                     ],
3876                                     "properties": {
3877                                         "fr":{
3878                                             "type": "boolean",
3879                                             "description": "Factory Reset"
3880                                         },
3881                                         "rb": {
3882                                             "type": "boolean",
3883                                             "description": "Reboot Action"
3884                                         }
3885                                     }
3886                                 },
3887                                 "type": "object",
3888                                 "allOf": [
3889                                     { "$ref": "oic.core-schema.json#/definitions/oic.core"},
3890                                     { "$ref": "#/definitions/oic.wk.mnt" }
3891                                 ]
3892                             }
3893
3894                     example: /
3895                         {
3896                             "rt":  ["oic.wk.mnt"],
3897                             "fr":  false,
3898                             "rb":  false
3899                         }
3900
3901 post:
3902     description: |
3903         Set the maintenance action(s)
3904
3905     is : ['interface-update']
3906
3907     body:
3908         application/json:
3909             schema: /
3910                 {
3911                     "$schema": "http://json-schemas.org/draft-04/schema#",
3912                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights

```

```

3912 reserved.",
3913     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-schema.json#",
3914     "definitions": {
3915         "oic.wk.mnt": {
3916             "type": "object",
3917             "anyOf": [
3918                 {"required": ["fr"]},
3919                 {"required": ["rb"]}
3920             ],
3921             "properties": {
3922                 "fr": {
3923                     "type": "boolean",
3924                     "description": "Factory Reset"
3925                 },
3926                 "rb": {
3927                     "type": "boolean",
3928                     "description": "Reboot Action"
3929                 }
3930             }
3931         }
3932     },
3933     "type": "object",
3934     "allOf": [
3935         {"$ref": "oic.core-schema.json#/definitions/oic.core"},
3936         {"$ref": "#/definitions/oic.wk.mnt" }
3937     ]
3938 }
3939
3940 example: /
3941 {
3942     "n": "My Maintenance Actions",
3943     "fr": false,
3944     "rb": false
3945 }
3946
3947 responses :
3948 200:
3949     body:
3950     application/json:
3951         schema: /
3952         {
3953             "$schema": "http://json-schemas.org/draft-04/schema#",
3954             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
reserved.",
3955             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.mnt-
schema.json#",
3956             "definitions": {
3957                 "oic.wk.mnt": {
3958                     "type": "object",
3959                     "anyOf": [
3960                         {"required": ["fr"]},
3961                         {"required": ["rb"]}
3962                     ],
3963                     "properties": {
3964                         "fr": {
3965                             "type": "boolean",
3966                             "description": "Factory Reset"
3967                         },
3968                         "rb": {
3969                             "type": "boolean",
3970                             "description": "Reboot Action"
3971                         }
3972                     }
3973                 }
3974             }
3975         },
3976         "type": "object",
3977         "allOf": [
3978

```

```

3979         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
3980         { "$ref": "#/definitions/oic.wk.mnt" }
3981     ]
3982 }
3983
3984     example: /
3985     {
3986         "n": "My Maintenance Actions",
3987         "fr": false,
3988         "rb": false
3989     }
3990

```

### 3991 D.5.5 Property Definition

| Property name | Value type | Mandatory | Access mode | Description   |
|---------------|------------|-----------|-------------|---------------|
| fr            | boolean    | yes       | Read Write  | Factory Reset |
| rb            | boolean    | yes       | Read Write  | Reboot Action |

### 3992 D.5.6 CRUDN behavior

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

## 3993 D.6 Platform

### 3994 D.6.1 Introduction

3995 Known resource that is defines the platform on which an Server is hosted. Allows for platform  
3996 specific information to be discovered.

### 3997 D.6.2 Wellknown URI

3998 /oic/p

### 3999 D.6.3 Resource Type

4000 The resource type (rt) is defined as: oic.wk.p.

### 4001 D.6.4 RAML Definition

```

4002 #%RAML 0.8
4003 title: Platform
4004 version: v1-20160622
4005 traits:
4006   - interface :
4007     queryParameters:
4008       if:
4009         enum: ["oic.if.r", "oic.if.baseline"]
4010
4011 /oic/p:
4012   description: |
4013     Known resource that is defines the platform on which an Server is hosted.
4014     Allows for platform specific information to be discovered.
4015
4016   is : ['interface']
4017   get:
4018     description: |
4019       Retrieve the information about the Platform
4020
4021   responses :
4022     200:
4023       body:

```

```

4024     application/json:
4025         schema: /
4026             {
4027                 "$schema": "http://json-schemas.org/draft-04/schema#",
4028                 "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4029 reserved.",
4030                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.p-
4031 schema.json#",
4032                 "definitions": {
4033                     "oic.wk.p": {
4034                         "type": "object",
4035                         "properties": {
4036                             "pi": {
4037                                 "$ref": "oic.types-schema.json#/definitions/uuid",
4038                                 "readOnly": true,
4039                                 "description": "Platform Identifier as a UUID"
4040                             },
4041                             "mnmn": {
4042                                 "type": "string",
4043                                 "readOnly": true,
4044                                 "description": "Manufacturer Name",
4045                                 "maxLength": 64
4046                             },
4047                             "mnml": {
4048                                 "type": "string",
4049                                 "readOnly": true,
4050                                 "description": "Manufacturer's URL",
4051                                 "maxLength": 256,
4052                                 "format": "uri"
4053                             },
4054                             "mnmo": {
4055                                 "type": "string",
4056                                 "maxLength": 64,
4057                                 "readOnly": true,
4058                                 "description": "Model number as designated by manufacturer"
4059                             },
4060                             "mndt": {
4061                                 "$ref": "oic.types-schema.json#/definitions/date",
4062                                 "readOnly": true,
4063                                 "description": "Manufacturing Date."
4064                             },
4065                             "mnpv": {
4066                                 "type": "string",
4067                                 "maxLength": 64,
4068                                 "readOnly": true,
4069                                 "description": "Platform Version"
4070                             },
4071                             "mnos": {
4072                                 "type": "string",
4073                                 "maxLength": 64,
4074                                 "readOnly": true,
4075                                 "description": "Platform Resident OS Version"
4076                             },
4077                             "mnhw": {
4078                                 "type": "string",
4079                                 "maxLength": 64,
4080                                 "readOnly": true,
4081                                 "description": "Platform Hardware Version"
4082                             },
4083                             "mnfv": {
4084                                 "type": "string",
4085                                 "maxLength": 64,
4086                                 "readOnly": true,
4087                                 "description": "Manufacturer's firmware version"
4088                             },
4089                             "mnsl": {
4090                                 "type": "string",
4091                                 "readOnly": true,
4092                                 "description": "Manufacturer's Support Information URL",
4093                                 "maxLength": 256,

```

```

4094         "format": "uri"
4095     },
4096     "st": {
4097         "type": "string",
4098         "readOnly": true,
4099         "description": "Reference time for the device as defined in ISO 8601, where
4100 concatenation of 'date' and 'time' with the 'T' as a delimiter between 'date' and 'time'. The
4101 format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z.",
4102         "format": "date-time"
4103     },
4104     "vid": {
4105         "type": "string",
4106         "maxLength": 64,
4107         "readOnly": true,
4108         "description": "Manufacturer's defined string for the platform. The string
4109 is freeform and up to the manufacturer on what text to populate it"
4110     }
4111 }
4112 },
4113 },
4114 "type": "object",
4115 "allOf": [
4116     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4117     { "$ref": "#/definitions/oic.wk.p" }
4118 ],
4119 "required": [ "pi", "mnmn" ]
4120 }
4121
4122 example: /
4123 {
4124     "pi": "54919CA5-4101-4AE4-595B-353C51AA983C",
4125     "rt": ["oic.wk.p"],
4126     "mnmn": "Acme, Inc"
4127 }
4128

```

#### 4129 D.6.5 Property Definition

| Property name | Value type                    | Mandatory | Access mode | Description   |
|---------------|-------------------------------|-----------|-------------|---|
| mnfv          | string                        |           | Read Only   | Manufacturer's firmware version   |
| vid           | string                        |           | Read Only   | Manufacturer's defined string for the platform. The string is freeform and up to the manufacturer on what text to populate it |
| mnmn          | string                        | yes       | Read Only   | Manufacturer Name   |
| mnmo          | string                        |           | Read Only   | Model number as designated by manufacturer  |
| mnml          | string                        |           | Read Only   | Manufacturer's URL  |
| mnos          | string                        |           | Read Only   | Platform Resident OS Version  |
| mndt          | multiple types:<br>see schema |           | Read Only   | Manufacturing Date.   |
| st            | string                        |           | Read Only   | Reference time for the device as defined in ISO 8601, where concatenation of 'date' and 'time' with the 'T' as a              |

|      |                               |     |           |  |
|------|-------------------------------|-----|-----------|--|
|      |                               |     |           | delimiter between 'date' and 'time'. The format is [yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]Z. |
| mnsI | string                        |     | Read Only | Manufacturer's Support Information URL   |
| mnpv | string                        |     | Read Only | Platform Version   |
| pi   | multiple types:<br>see schema | yes | Read Only | Platform Identifier as a UUID  |
| mnhw | string                        |     | Read Only | Platform Hardware Version  |

4130 **D.6.6 CRUDN behavior**

| Resource | Create | Read | Update | Delete | Notify |
|----------|--------|------|--------|--------|--------|
| /oic/p   |        | get  |        |        |        |

4131 **D.7 Ping**

4132 **D.7.1 Introduction**

4133 The resource using which an Client keeps its Connection with an Server active.

4134 **D.7.2 Wellknown URI**

4135 /oic/ping

4136 **D.7.3 Resource Type**

4137 The resource type (rt) is defined as: oic.wk.ping.

4138 **D.7.4 RAML Definition**

```

4139 #%RAML 0.8
4140 title: Ping
4141 version: v1-20160622
4142 traits:
4143   - interface :
4144     queryParameters:
4145       if:
4146         enum: ["oic.if.rw", "oic.if.baseline"]
4147
4148 /oic/ping:
4149   description: |
4150     The resource using which an Client keeps its Connection with an Server active.
4151
4152   is : ['interface']
4153   get:
4154     description: |
4155       Retrieve the ping information
4156
4157   responses :
4158     200:
4159       body:
4160         application/json:
4161           schema: /
4162           {
4163             "$schema": "http://json-schemas.org/draft-04/schema#",

```



```

4164         "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4165 reserved.",
4166         "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.ping-
4167 schema.json#",
4168         "definitions": {
4169             "oic.wk.ping": {
4170                 "type": "object",
4171                 "properties": {
4172                     "in": {
4173                         "type": "integer",
4174                         "description": "ReadWrite, Indicates the interval for which connection
4175 shall be kept alive"
4176                     }
4177                 }
4178             },
4179             "type": "object",
4180             "allOf": [
4181                 { "$ref": "oic.core-schema.json#/definitions/oic.core"},
4182                 { "$ref": "#/definitions/oic.wk.ping"}
4183             ],
4184             "required": [
4185                 "in"
4186             ]
4187         }
4188     }
4189
4190     example: /
4191     {
4192         "rt": ["oic.wk.ping"],
4193         "n": "Ping Information",
4194         "in": 16
4195     }
4196

```

#### 4197 D.7.5 Property Definition

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

#### 4198 D.7.6 CRUDN behavior

| Resource  | Create | Read | Update | Delete | Notify |
|-----------|--------|------|--------|--------|--------|
| /oic/ping |        | get  |        |        |        |

### 4199 D.8 Discoverable Resources, baseline interface

#### 4200 D.8.1 Introduction

4201 Baseline representation of /oic/res; list of discoverable resources

#### 4202 D.8.2 Wellknown URI

4203 /oic/res

#### 4204 D.8.3 Resource Type

4205 The resource type (rt) is defined as: oic.wk.res.

#### 4206 D.8.4 RAML Definition

4207 `##RAML 0.8`

4208 `title: Discoverable Resources`

4209 `version: v1-20160622`

4210 `traits:`

```

4211 - interface-ll :
4212     queryParameters:
4213         if:
4214             enum: ["oic.if.ll"]
4215 - interface-baseline :
4216     queryParameters:
4217         if:
4218             enum: ["oic.if.baseline"]
4219
4220 /oic-res-baseline-URI:
4221     description: |
4222     Baseline representation of /oic/res; list of discoverable resources
4223
4224     is : ['interface-baseline']
4225     get:
4226         description: |
4227         Retrieve the discoverable resource set, baseline interface
4228
4229     responses :
4230         200:
4231             body:
4232                 application/json:
4233                     schema: /
4234                         {
4235                             "$schema": "http://json-schema.org/draft-v4/schema#",
4236                             "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4237 reserved.",
4238                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-
4239 schema.json#",
4240                             "definitions": {
4241                                 "oic.res-baseline": {
4242                                     "type": "object",
4243                                     "properties": {
4244                                         "rt": {
4245                                             "type": "array",
4246                                             "items" : {
4247                                                 "type" : "string",
4248                                                 "maxLength": 64
4249                                             },
4250                                             "minItems" : 1,
4251                                             "readOnly": true,
4252                                             "description": "Resource Type"
4253                                         },
4254                                         "if": {
4255                                             "type": "array",
4256                                             "items": {
4257                                                 "type" : "string",
4258                                                 "enum" : ["oic.if.baseline", "oic.if.ll"]
4259                                             },
4260                                             "minItems": 1,
4261                                             "readOnly": true,
4262                                             "description": "The interface set supported by this resource"
4263                                         },
4264                                         "n": {
4265                                             "type": "string",
4266                                             "maxLength": 64,
4267                                             "readOnly": true,
4268                                             "description": "Human friendly name"
4269                                         },
4270                                         "di": {
4271                                             "$ref": "oic.types-schema.json#/definitions/uuid",
4272                                             "readOnly": true,
4273                                             "description": "Unique identifier for device (UUID) as indicated by the

```

```

4274 /oic/d resource of the device"
4275     },
4276     "mpro": {
4277         "readOnly": true,
4278         "description": "Supported messaging protocols",
4279         "type": "string",
4280         "maxLength": 64
4281     },
4282     "links": {
4283         "type": "array",
4284         "items": {
4285             "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
4286         }
4287     },
4288     },
4289     "required": ["rt", "if", "di", "links"]
4290 }
4291 },
4292 "description": "The list of resources expressed as OIC links",
4293 "type": "array",
4294 "items": {
4295     "$ref": "#/definitions/oic.res-baseline"
4296 }
4297 }
4298
4299 example: /
4300 [
4301     {
4302         "rt": ["oic.wk.res"],
4303         "if": ["oic.if.baseline", "oic.if.ll" ],
4304         "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
4305         "links":
4306         [
4307             {
4308                 "href": "/humidity",
4309                 "rt": ["oic.r.humidity"],
4310                 "if": ["oic.if.s"]
4311             },
4312             {
4313                 "href": "/temperature",
4314                 "rt": ["oic.r.temperature"],
4315                 "if": ["oic.if.s"]
4316             }
4317         ]
4318     }
4319 ]
4320

```

### 4321 D.8.5 Property Definition

| Property name | Value type                 | Mandatory | Access mode | Description   |
|---------------|----------------------------|-----------|-------------|---|
| rt            | array: see schema          | yes       | Read Only   | Resource Type   |
| links         | array: see schema          | yes       | Read Write  |   |
| di            | multiple types: see schema | yes       | Read Only   | Unique identifier for device (UUID) as indicated by the /oic/d resource of the device |
| mpro          | string                     |           | Read Only   | Supported messaging protocols   |
| n             | string                     |           | Read Only   | Human friendly name   |

|    |                  |     |     |           |  |
|----|------------------|-----|-----|-----------|--|
| if | array:<br>schema | see | yes | Read Only | The interface set supported by this resource |
|----|------------------|-----|-----|-----------|--|

4322 **D.8.6 CRUDN behavior**

| Resource | Create | Read | Update | Delete | Notify |
|----------|--------|------|--------|--------|--------|
| /oic/res |        | get  |        |        |        |

4323 **D.9 Discoverable Resources, link list interface**

4324 **D.9.1 Introduction**

4325 Link list representation of /oic/res; list of discoverable resources

4326 **D.9.2 Wellknown URI**

4327 /oic/res

4328 **D.9.3 Resource Type**

4329 The resource type (rt) is defined as: oic.wk.res.

4330 **D.9.4 RAML Definition**

```

4331 #%RAML 0.8
4332 title: Discoverable Resources
4333 version: v1-20160622
4334 traits:
4335   - interface-ll :
4336     queryParameters:
4337       if:
4338         enum: ["oic.if.ll"]
4339   - interface-baseline :
4340     queryParameters:
4341       if:
4342         enum: ["oic.if.baseline"]
4343
4344 /oic-res-ll-URI:
4345   description: |
4346     Link list representation of /oic/res; list of discoverable resources
4347
4348   is : ['interface-ll']
4349   get:
4350     description: |
4351       Retrieve the discoverable resource set, link list interface
4352
4353   responses :
4354     200:
4355       body:
4356         application/json:
4357           schema: /
4358             {
4359               "$schema": "http://json-schema.org/draft-v4/schema#",
4360               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4361 reserved.",
4362               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.wk.res-schema-
4363 ll.json#",
4364               "definitions": {
4365                 "oic.res-ll": {
4366                   "type": "object",
4367                   "properties": {

```

```

4368         "di": {
4369             "$ref": "oic.types-schema.json#/definitions/uuid",
4370             "readOnly": true,
4371             "description": "Unique identifier for device (UUID) as indicated by the
4372 /oic/d resource of the device"
4373         },
4374         "links": {
4375             "type": "array",
4376             "items": {
4377                 "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"
4378             }
4379         },
4380     },
4381     "required": ["di", "links"]
4382 }
4383 },
4384 "description": "The list of resources expressed as OIC links with di ",
4385 "type": "array",
4386 "items": {
4387     "$ref": "#/definitions/oic.res-ll"
4388 }
4389 }
4390
4391 example: /
4392 [
4393     {
4394         "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
4395         "links":
4396             [
4397                 {
4398                     "href": "/humidity",
4399                     "rt": ["oic.r.humidity"],
4400                     "if": ["oic.if.s"]
4401                 },
4402                 {
4403                     "href": "/temperature",
4404                     "rt": ["oic.r.temperature"],
4405                     "if": ["oic.if.s"]
4406                 }
4407             ]
4408     }
4409 ]
4410

```

### D.9.5 Property Definition

| Property name | Value type                 | Mandatory | Access mode | Description   |
|---------------|----------------------------|-----------|-------------|---|
| links         | array: see schema          | yes       | Read Write  |   |
| di            | multiple types: see schema | yes       | Read Only   | Unique identifier for device (UUID) as indicated by the /oic/d resource of the device |
| rt            | array: see schema          | yes       | Read Write  | Resource Type   |
| di            | multiple types: see schema |           | Read Write  | Unique identifier for device (UUID)   |
| title         | string                     |           | Read Write  | A title for the link relation. Can be used by the UI to provide a context             |
| huri          | string                     |           | Read Write  | The base URI used to fully  |

|        |                               |     |            |  |
|--------|-------------------------------|-----|------------|--|
|        |                               |     |            | qualify a Relative Reference in the href parameter. Use the OCF Schema for URI   |
| ins    | multiple types:<br>see schema |     | Read Write | The instance identifier for this web link in an array of web links - used in collections   |
| p      | object: see<br>schema         |     | Read Write | Specifies the framework policies on the Resource referenced by the target URI  |
| href   | string                        | yes | Read Write | This is the target URI, it can be specified as a Relative Reference or fully-qualified URI. Relative Reference should be used along with the di parameter to make it unique. |
| rel    | multiple types:<br>see schema |     | Read Write | The relation of the target URI referenced by the link to the context URI   |
| type   | array: see<br>schema          |     | Read Write | A hint at the representation of the resource referenced by the target URI. This represents the media types that are used for both accepting and emitting                     |
| anchor | string                        |     | Read Write | This is used to override the context URI e.g. override the URI of the containing collection  |
| if     | array: see<br>schema          | yes | Read Write | The interface set supported by this resource   |

4412

#### D.9.6 CRUDN behavior

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

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

4413 **D.9.7 Referenced JSON schemas**

4414 **D.9.8 oic.oic-link-schema.json**

```

4415 {
4416   "$schema": "http://json-schema.org/draft-04/schema#",
4417   "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights reserved.",
4418   "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.oic-link-schema.json#",
4419   "definitions": {
4420     "oic.oic-link": {
4421       "type": "object",
4422       "properties": {
4423         "href": {
4424           "type": "string",
4425           "maxLength": 256,
4426           "description": "This is the target URI, it can be specified as a Relative Reference or
4427 fully-qualified URI. Relative Reference should be used along with the di parameter to make it
4428 unique.",
4429           "format": "uri"
4430         },
4431         "rel": {
4432           "oneOf": [
4433             {
4434               "type": "array",
4435               "items": {
4436                 "type": "string",
4437                 "maxLength": 64
4438               },
4439               "minItems": 1,
4440               "default": ["hosts"]
4441             },
4442             {
4443               "type": "string",
4444               "maxLength": 64,
4445               "default": "hosts"
4446             }
4447           ],
4448           "description": "The relation of the target URI referenced by the link to the context URI"
4449         },
4450         "rt": {
4451           "type": "array",
4452           "items": {
4453             "type": "string",
4454             "maxLength": 64
4455           },
4456           "minItems": 1,
4457           "description": "Resource Type"
4458         },
4459         "if": {
4460           "type": "array",
4461           "items": {
4462             "type": "string",
4463             "enum": ["oic.if.baseline", "oic.if.ll", "oic.if.b", "oic.if.rw", "oic.if.r",
4464 "oic.if.a", "oic.if.s" ]
4465           },
4466           "minItems": 1,
4467           "description": "The interface set supported by this resource"
4468         },
4469         "di": {
4470           "$ref": "oic.types-schema.json#/definitions/uuid",
4471           "description": "Unique identifier for device (UUID)"
4472         },
4473         "buri": {
4474           "type": "string",
4475           "description": "The base URI used to fully qualify a Relative Reference in the href
4476 parameter. Use the OCF Schema for URI",
4477           "maxLength": 256,
4478           "format": "uri"
4479         },

```

```

4480     "p": {
4481         "description": "Specifies the framework policies on the Resource referenced by the target
4482 URI",
4483         "type": "object",
4484         "properties": {
4485             "bm": {
4486                 "description": "Specifies the framework policies on the Resource referenced by the
4487 target URI for e.g. observable and discoverable",
4488                 "type": "integer"
4489             },
4490             "sec": {
4491                 "description": "Specifies if security needs to be turned on when looking to interact
4492 with the Resource",
4493                 "default": false,
4494                 "type": "boolean"
4495             },
4496             "port": {
4497                 "description": "Secure port to be used for connection",
4498                 "type": "integer"
4499             }
4500         },
4501         "required" : ["bm"]
4502     },
4503     "title": {
4504         "type": "string",
4505         "maxLength": 64,
4506         "description": "A title for the link relation. Can be used by the UI to provide a
4507 context"
4508     },
4509     "anchor": {
4510         "type": "string",
4511         "maxLength": 256,
4512         "description": "This is used to override the context URI e.g. override the URI of the
4513 containing collection",
4514         "format": "uri"
4515     },
4516     "ins": {
4517         "oneOf": [
4518             {
4519                 "type": "integer",
4520                 "description": "An ordinal number that is not repeated - must be unique in the
4521 collection context"
4522             },
4523             {
4524                 "type": "string",
4525                 "maxLength": 256,
4526                 "format" : "uri",
4527                 "description": "Any unique string including a URI"
4528             },
4529             {
4530                 "$ref": "oic.types-schema.json#/definitions/uuid",
4531                 "description": "Unique identifier (UUID)"
4532             }
4533         ],
4534         "description": "The instance identifier for this web link in an array of web links - used
4535 in collections"
4536     },
4537     "type": {
4538         "type": "array",
4539         "description": "A hint at the representation of the resource referenced by the target
4540 URI. This represents the media types that are used for both accepting and emitting",
4541         "items" : {
4542             "type": "string",
4543             "maxLength": 64
4544         },
4545         "minItems": 1,
4546         "default": "application/cbor"
4547     }
4548 },
4549 "required": [ "href", "rt", "if" ]
4550 }

```



```

4551     },
4552     "type": "object",
4553     "allOf": [
4554       { "$ref": "#/definitions/oic.oic-link" }
4555     ]
4556   }
4557 }

```

## 4558 D.10 Scenes (Top level)

### 4559 D.10.1 Introduction

4560 Toplevel Scene resource. This resource is a generic collection resource. The rts value shall contain  
 4561 oic.scenecollection resource types.

### 4562 D.10.2 Example URI

4563 /SceneListResURI

### 4564 D.10.3 Resource Type

4565 The resource type (rt) is defined as: oic.wk.scenelist.

### 4566 D.10.4 RAML Definition

```

4567 #%RAML 0.8
4568 title: Scene
4569 version: v1-20160622
4570 traits:
4571   - interface :
4572     queryParameters:
4573       if:
4574         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
4575
4576 /SceneListResURI:
4577   description: |
4578     Toplevel Scene resource.
4579     This resource is a generic collection resource.
4580     The rts value shall contain oic.scenecollection resource types.
4581
4582   get:
4583     description: |
4584       Provides the current list of web links pointing to scenes
4585
4586     responses :
4587       200:
4588         body:
4589           application/json:
4590             schema: /
4591             {
4592               "$schema": "http://json-schema.org/draft-04/schema#",
4593               "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
4594 reserved.",
4595               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.collection-
4596 schema.json#",
4597               "title": "Collection",
4598               "definitions": {
4599                 "oic.collection.setoflinks": {
4600                   "description": "A set (array) of simple or individual OIC Links. In
4601 addition to properties required for an OIC Link, the identifier for that link in this set is also
4602 required",
4603                   "type": "array",
4604                   "items": {
4605                     "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link"

```

```

4606     }
4607   },
4608   "oic.collection.alllinks": {
4609     "description": "All forms of links in a collection",
4610     "oneOf": [
4611       {
4612         "$ref": "#/definitions/oic.collection.setoflinks"
4613       }
4614     ]
4615   },
4616   "oic.collection": {
4617     "type": "object",
4618     "description": "A collection is a set (array) of tagged-link or set
4619 (array) of simple links along with additional properties to describe the collection itself",
4620     "properties": {
4621       "n": {
4622         "type": "string",
4623         "description": "User friendly name of the
4624 collection"
4625       },
4626       "id": {
4627         "anyOf": [
4628           {
4629             "type": "integer",
4630             "description": "A number that is unique to that
4631 collection; like an ordinal number that is not repeated"
4632           },
4633           {
4634             "type": "string",
4635             "description": "A unique string that could be a hash or
4636 similarly unique"
4637           },
4638           {
4639             "$ref": "oic.types-schema.json#/definitions/uuid",
4640             "description": "A unique string that could be a UUIDv4"
4641           }
4642         ],
4643         "description": "ID for the collection. Can be an value that is
4644 unique to the use context or a UUIDv4"
4645       },
4646       "di": {
4647         "$ref": "oic.types-schema.json#/definitions/uuid",
4648         "description": "The device ID which is an UUIDv4 string; used for
4649 backward compatibility with Spec A definition of /oic/res"
4650       },
4651       "rts": {
4652         "$ref": "oic.core-
4653 schema.json#/definitions/oic.core/properties/rt",
4654         "description": "Defines the list of allowable resource types (for
4655 Target and anchors) in links included in the collection; new links being created can only be from
4656 this list"
4657       },
4658       "drel": {
4659         "type": "string",
4660         "description": "When specified this is the default relationship
4661 to use when an OIC Link does not specify an explicit relationship with *rel* parameter"
4662       },
4663       "links": {
4664         "$ref": "#/definitions/oic.collection.alllinks"
4665       }
4666     }
4667   },
4668   "type": "object",
4669   "allOf": [
4670     {"$ref": "oic.core-schema.json#/definitions/oic.core"},
4671     {"$ref": "#/definitions/oic.collection"}
4672   ]
4673 }
4674
4675 example: /

```

```

4675     {
4676         "rt":      ["oic.wk.scenelist"],
4677         "n":      "list of scene Collections",
4678         "rts":    ["oic.wk.scenecollection"],
4679         "links": [
4680             ]
4681     }
4682

```

4683 **D.10.5 Property Definition**

| Property name | Value type                    | Mandatory | Access mode | Description   |
|---------------|-------------------------------|-----------|-------------|---|
| links         | multiple types:<br>see schema |           | Read Write  |   |
| di            | multiple types:<br>see schema |           | Read Write  | The device ID which is an UUIDv4 string; used for backward compatibility with Spec A definition of /oic/res   |
| n             | string                        |           | Read Write  | User friendly name of the collection  |
| drel          | string                        |           | Read Write  | When specified this is the default relationship to use when an OIC Link does not specify an explicit relationship with *rel* parameter                        |
| rts           | multiple types:<br>see schema |           | Read Write  | Defines the list of allowable resource types (for Target and anchors) in links included in the collection; new links being created can only be from this list |
| id            | multiple types:<br>see schema |           | Read Write  | ID for the collection. Can be an value that is unique to the use context or a UUIDv4  |

4684 **D.10.6 CRUDN behavior**

| Resource         | Create | Read | Update | Delete | Notify |
|------------------|--------|------|--------|--------|--------|
| /SceneListResURI |        | get  |        |        |        |

## 4685 D.11 Scene Collections

### 4686 D.11.1 Introduction

4687 Collection that models a set of Scenes. This resource is a generic collection resource with  
4688 additional parameters. The rts value shall contain oic.scenemember resource types. The additional  
4689 parameters are: lastScene, this is the scene value last set by any OIC Client, sceneValues, this  
4690 is the list of available scenes, lastScene shall be listed in sceneValues.

### 4691 D.11.2 Example URI

4692 /SceneCollectionResURI

### 4693 D.11.3 Resource Type

4694 The resource type (rt) is defined as: oic.wk.scenecollection.

### 4695 D.11.4 RAML Definition

```
4696 #%RAML 0.8
4697 title: Scene
4698 version: v1-20160622
4699 traits:
4700   - interface :
4701     queryParameters:
4702       if:
4703         enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
4704
4705 /SceneCollectionResURI:
4706   description: |
4707     Collection that models a set of Scenes.
4708     This resource is a generic collection resource with additional parameters.
4709     The rts value shall contain oic.scenemember resource types.
4710     The additional parameters are:
4711     lastScene, this is the scene value last set by any OIC Client,
4712     sceneValues, this is the list of available scenes,
4713     lastScene shall be listed in sceneValues.
4714
4715   get:
4716     description: |
4717       Provides the current list of web links pointing to scenes
4718
4719     responses :
4720       200:
4721         body:
4722           application/json:
4723             schema: /
4724             {
4725               "$schema": "http://json-schema.org/draft-04/schema#",
4726               "description" : "Copyright (c) 2016-2017 Open Connectivity Foundation, Inc. All
4727 rights reserved.",
4728               "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
4729 schema.json#",
4730               "title" : "Scene Collection",
4731               "definitions": {
4732                 "oic.sceneCollection": {
4733                   "type": "object",
4734                   "properties": {
4735                     "lastScene": {
4736                       "type": "string",
4737                       "description": "Last selected Scene, shall be part of sceneValues"
4738                     },
4739                     "sceneValues": {
4740                       "type": "string",
```

```

4741         "readOnly": true,
4742         "description": "All available scene values"
4743     },
4744     "n": {
4745         "type": "string",
4746         "description": "Used to name the Scene collection"
4747     },
4748     "id": {
4749         "type": "string",
4750         "description": "A unique string that could be a hash or
4751 similarly unique"
4752     },
4753     "rts": {
4754         "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
4755         "description": "Defines the list of allowable resource types in links
4756 included in the collection; new links being created can only be from this list"
4757     },
4758     "links": {
4759         "type": "array",
4760         "description": "Array of OIC web links that are reference from this
4761 collection",
4762         "items": {
4763             "allOf": [
4764                 { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
4765                 { "required": [ "ins" ] }
4766             ]
4767         }
4768     },
4769     },
4770     "required": [ "lastScene", "sceneValues", "rts", "id" ]
4771 }
4772 },
4773
4774 "type": "object",
4775 "allOf": [
4776     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4777     { "$ref": "#/definitions/oic.sceneCollection" }
4778 ]
4779 }
4780
4781 example: /
4782 {
4783     "lastScene": "off",
4784     "sceneValues": "off,Reading,TVWatching",
4785     "rt":         ["oic.wk.scenecollection"],
4786     "n":         "My Scenes for my living room",
4787     "id":        "0685B960-736F-46F7-BECO-9E6CBD671ADC1",
4788     "rts":       ["oic.wk.scenemember"],
4789     "links": [
4790         ]
4791     }
4792
4793 post:
4794 description: |
4795     Provides the action to change the last set scene selection.
4796     Calling this method shall update of all scene members to the prescribed membervalue.
4797     When this method is called with the same value as the current lastScene value
4798     then all scene members shall be updated.
4799
4800 body:
4801 application/json:
4802 schema: /
4803 {
4804     "$schema": "http://json-schema.org/draft-04/schema#",
4805     "description": "Copyright (c) 2016-2017 Open Connectivity Foundation, Inc. All rights
4806 reserved.",
4807     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-

```

```

4808 schema.json#",
4809     "title" : "Scene Collection",
4810     "definitions": {
4811         "oic.sceneCollection": {
4812             "type": "object",
4813             "properties": {
4814                 "lastScene": {
4815                     "type": "string",
4816                     "description": "Last selected Scene, shall be part of sceneValues"
4817                 },
4818                 "sceneValues": {
4819                     "type": "string",
4820                     "readOnly": true,
4821                     "description": "All available scene values"
4822                 },
4823                 "n": {
4824                     "type": "string",
4825                     "description": "Used to name the Scene collection"
4826                 },
4827                 "id": {
4828                     "type": "string",
4829                     "description" : "A unique string that could be a hash or
4830 similarly unique"
4831                 },
4832                 "rts": {
4833                     "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
4834                     "description": "Defines the list of allowable resource types in links included
4835 in the collection; new links being created can only be from this list"
4836                 },
4837                 "links": {
4838                     "type": "array",
4839                     "description": "Array of OIC web links that are reference from this
4840 collection",
4841                     "items" : {
4842                         "allOf": [
4843                             { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
4844                             { "required" : [ "ins" ] }
4845                         ]
4846                     }
4847                 },
4848                 "required": [ "lastScene" ]
4849             }
4850         },
4851     },
4852     "type": "object",
4853     "allOf" : [
4854         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4855         { "$ref": "#/definitions/oic.sceneCollection" }
4856     ]
4857 }
4858 }
4859
4860 example: /
4861 {
4862     "lastScene": "Reading"
4863 }
4864
4865 responses :
4866 200:
4867     description: |
4868     Indicates that the value is changed.
4869     The changed properties are provided in the response.
4870
4871     body:
4872     application/json:
4873     schema: /

```

```

4874     {
4875         "$schema": "http://json-schema.org/draft-04/schema#",
4876         "description": "Copyright (c) 2016-2017 Open Connectivity Foundation, Inc. All
rights reserved.",
4877     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneCollection-
4878     schema.json#",
4879     "title": "Scene Collection",
4880     "definitions": {
4881         "oic.sceneCollection": {
4882             "type": "object",
4883             "properties": {
4884                 "lastScene": {
4885                     "type": "string",
4886                     "description": "Last selected Scene, shall be part of sceneValues"
4887                 },
4888                 "sceneValues": {
4889                     "type": "string",
4890                     "readOnly": true,
4891                     "description": "All available scene values"
4892                 },
4893                 "n": {
4894                     "type": "string",
4895                     "description": "Used to name the Scene collection"
4896                 },
4897                 "id": {
4898                     "type": "string",
4899                     "description": "A unique string that could be a hash or
4900     similarly unique"
4901                 },
4902                 "rts": {
4903                     "$ref": "oic.core-schema.json#/definitions/oic.core/properties/rt",
4904                     "description": "Defines the list of allowable resource types in links
4905     included in the collection; new links being created can only be from this list"
4906                 },
4907                 "links": {
4908                     "type": "array",
4909                     "description": "Array of OIC web links that are reference from this
4910     collection",
4911                 "items": {
4912                     "allOf": [
4913                         { "$ref": "oic.oic-link-schema.json#/definitions/oic.oic-link" },
4914                         { "required": [ "ins" ] }
4915                     ]
4916                 }
4917             }
4918         },
4919         "required": [ "lastScene" ]
4920     },
4921     },
4922     "type": "object",
4923     "allOf": [
4924         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
4925         { "$ref": "#/definitions/oic.sceneCollection" }
4926     ]
4927 }
4928 }
4929 }
4930
4931     example: /
4932     {
4933         "lastScene": "Reading"
4934     }
4935

```

#### 4936 D.11.5 Property Definition

| Property name | Value type | Mandatory | Access mode | Description                   |
|---------------|------------|-----------|-------------|-------------------------------|
| lastScene     | string     | yes       | Read Write  | Last selected Scene, shall be |

|             |                            |     |            |  |
|-------------|----------------------------|-----|------------|--|
|             |                            |     |            | part of sceneValues  |
| links       | array: see schema          |     | Read Write | Array of OIC web links that are reference from this collection   |
| sceneValues | string                     | yes | Read Only  | All available scene values   |
| n           | string                     |     | Read Write | Used to name the Scene collection  |
| rts         | multiple types: see schema | yes | Read Write | Defines the list of allowable resource types in links included in the collection; new links being created can only be from this list |
| id          | string                     | yes | Read Write | A unique string that could be a hash or similarly unique   |

4937 **D.11.6 CRUDN behavior**

| Resource               | Create | Read | Update | Delete | Notify |
|------------------------|--------|------|--------|--------|--------|
| /SceneCollectionResURI |        | get  | post   |        |        |

4938 **D.12 Scene Member**

4939 **D.12.1 Introduction**

4940 Collection that models a scene member.

4941 **D.12.2 Example URI**

4942 /SceneMemberResURI

4943 **D.12.3 Resource Type**

4944 The resource type (rt) is defined as: oic.wk.scenemember.

4945 **D.12.4 RAML Definition**

```

4946 #%RAML 0.8
4947 title: Scene
4948 version: v1-20160622
4949 traits:
4950 - interface :
4951     queryParameters:
4952         if:
4953             enum: ["oic.if.a", "oic.if.ll", "oic.if.baseline"]
4954
4955 /SceneMemberResURI:
4956     description: |
4957         Collection that models a scene member.
4958
4959     get:
4960         description: |

```



```

4961         Provides the scene member
4962
4963     responses :
4964         200:
4965             body:
4966                 application/json:
4967                     schema: /
4968                         {
4969                             "$schema": "http://json-schema.org/draft-04/schema#",
4970                             "description" : "Copyright (c) 2016-2017 Open Connectivity Foundation, Inc. All
4971 rights reserved.",
4972                             "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.sceneMember-
4973 schema.json#",
4974                             "title" : "Scene Member",
4975                             "definitions": {
4976                                 "oic.sceneMember": {
4977                                     "type": "object",
4978                                     "properties": {
4979                                         "n": {
4980                                             "type": "string",
4981                                             "description": "Used to name the Scene collection"
4982                                         },
4983                                         "id": {
4984                                             "type": "string",
4985                                             "description": "Can be an value that is unique to the use context or a
4986 UUIDv4"
4987                                         },
4988                                         "SceneMappings" : {
4989                                             "type": "array",
4990                                             "description": "array of mappings per scene, can be 1",
4991                                             "items": {
4992                                                 "type": "object",
4993                                                 "properties": {
4994                                                     "scene": {
4995                                                         "type": "string",
4996                                                         "description": "Specifies a scene value that will acted upon"
4997                                                     },
4998                                                     "memberProperty": {
4999                                                         "type": "string",
5000                                                         "readOnly": true,
5001                                                         "description": "property name that will be mapped"
5002                                                     },
5003                                                     "memberValue": {
5004                                                         "type": "string",
5005                                                         "readOnly": true,
5006                                                         "description": "value of the Member Property"
5007                                                     }
5008                                                 },
5009                                                 "required": [ "scene", "memberProperty", "memberValue" ]
5010                                             }
5011                                         },
5012                                         "link": {
5013                                             "type": "string",
5014                                             "description": "web link that points at an resource",
5015                                             "$ref": "oic.oic-link-schema.json#"
5016                                         }
5017                                     },
5018                                     "required": [ "link" ]
5019                                 }
5020                             },
5021                             "type": "object",
5022                             "allof" : [
5023                                 { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5024                                 { "$ref": "#/definitions/oic.sceneMember" }
5025                             ]
5026                         }
5027

```

```

5028     }
5029
5030     example: /
5031     {
5032         "rt": ["oic.wk.scenemember"],
5033         "id": "0685B960-FFFF-46F7-BEC0-9E6234671ADC1",
5034         "n": "my binary switch (for light bulb) mappings",
5035         "link": {
5036             "href": "binarySwitch",
5037             "rt": ["oic.r.switch.binary"],
5038             "if": ["oic.if.a", "oic.if.baseline"],
5039             "eps": [
5040                 {"ep": "coap://[fe80::b1d6]:1111", "pri": 2},
5041                 {"ep": "coaps://[fe80::b1d6]:1122"},
5042                 {"ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3}
5043             ]
5044         },
5045         "sceneMappings": [
5046             {
5047                 "scene": "off",
5048                 "memberProperty": "value",
5049                 "memberValue": true
5050             },
5051             {
5052                 "scene": "Reading",
5053                 "memberProperty": "value",
5054                 "memberValue": false
5055             },
5056             {
5057                 "scene": "TVWatching",
5058                 "memberProperty": "value",
5059                 "memberValue": true
5060             }
5061         ]
5062     }
5063

```

## 5064 D.12.5 Property Definition

| Property name | Value type        | Mandatory | Access mode | Description   |
|---------------|-------------------|-----------|-------------|---|
| SceneMappings | array: see schema |           | Read Write  | array of mappings per scene, can be 1                         |
| link          | string            | yes       | Read Write  | web link that points at an resource                           |
| id            | string            |           | Read Write  | Can be an value that is unique to the use context or a UUIDv4 |
| n             | string            |           | Read Write  | Used to name the Scene collection                             |

## 5065 D.12.6 CRUDN behavior

| Resource           | Create | Read | Update | Delete | Notify |
|--------------------|--------|------|--------|--------|--------|
| /SceneMemberResURI |        | get  |        |        |        |

## 5066 D.13 Resource directory resource

### 5067 D.13.1 Introduction

5068 Resource to be exposed by any Device that can act as a Resource Directory.

```

5069 D.13.2 Wellknown URI
5070 /oic/rd
5071 D.13.3 Resource Type
5072 The resource type (rt) is defined as: oic.wk.rd.
5073 D.13.4 RAML Definition
5074 #%RAML 0.8
5075 title: Resource Directory
5076 version: v1-20160622
5077 traits:
5078   - rddefinterface :
5079     queryParameters:
5080       if:
5081         default: oic.if.baseline
5082         enum: ["oic.if.baseline"]
5083         type: string
5084         description: Interface is optional since there is only one interface supported for the
5085 Resource Type
5086 Both for RD selection and for publish
5087 Example: GET /oic/rd?if=oic.wk.baseline
5088
5089
5090 /oic/rd:
5091   description: |
5092     Resource to be exposed by any Device that can act as a Resource Directory.
5093
5094   get:
5095     description: |
5096       Get the attributes of the Resource Directory for selection purposes.
5097
5098     queryParameters:
5099       rt:
5100         enum: oic.wk.rd
5101         type: string
5102         description: Only one Resource Type is used for GET; RT is optional
5103 Example: GET /oic/rd?rt=oic.wk.rd
5104
5105         required: false
5106         is : ['rddefinterface']
5107     responses :
5108       200:
5109         description: |
5110           Respond with the selector criteria - either the set of attributes or the bias factor
5111
5112         body:
5113           application/json:
5114             schema: /
5115               {
5116                 "$schema": "http://json-schema.org/draft-04/schema#",
5117                 "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5118 reserved.",
5119                 "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.selection-
5120 schema.json#",
5121                 "title" : "RD Selection",
5122                 "definitions": {
5123                   "oic.rd.attributes": {
5124                     "type": "object",
5125                     "properties": {

```

```

5126         "di": {
5127             "$ref": "oic.types-schema.json#/definitions/uuid",
5128             "description": "A unique identifier for the Resource Directory - the same
5129 as the device ID of the RD"
5130         },
5131     },
5132     "sel": {
5133         "description": "Selection criteria that a device wanting to publish to any
5134 RD can use to choose this Resource Directory over others that are discovered",
5135         "oneOf": [
5136             {
5137                 "type": "object",
5138                 "properties": {
5139                     "pwr": {
5140                         "type": "string",
5141                         "enum": [ "ac", "batt", "safe" ],
5142                         "description": "A hint about how the RD is powered. If AC then this
5143 is stronger than battery powered. If source is reliable (safe) then appropriate mechanism for
5144 managing power failure exists"
5145                     },
5146                     "conn": {
5147                         "type": "string",
5148                         "enum": [ "wrld", "wrlds" ],
5149                         "description": "A hint about the networking connectivity of the RD.
5150 *wrld* if wired connected and *wrlds* if wireless connected."
5151                     },
5152                     "bw": {
5153                         "type": "string",
5154                         "description": "Qualitative bandwidth of the connection",
5155                         "enum": [ "high", "low", "lossy" ]
5156                     },
5157                     "mf": {
5158                         "type": "integer",
5159                         "description": "Memory factor - Ratio of available memory to total
5160 memory expressed as a percentage"
5161                     },
5162                     "load": {
5163                         "type": "array",
5164                         "items": {
5165                             "type": "number"
5166                         },
5167                         "minitems": 3,
5168                         "maxitems": 3,
5169                         "description": "Current load capacity of the RD. Expressed as a
5170 load factor 3-tuple (upto two decimal points each). Load factor is based on request processed in a
5171 1 minute, 5 minute window and 15 minute window"
5172                     }
5173                 }
5174             },
5175             {
5176                 "type": "integer",
5177                 "minimum": 0,
5178                 "maximum": 100,
5179                 "description": "A bias factor calculated by the Resource directory -
5180 the value is in the range of 0 to 100 - 0 implies that RD is not to be selected. Client chooses RD
5181 with highest bias factor or randomly between RDs that have same bias factor"
5182             }
5183         ]
5184     }
5185 }
5186 }
5187 },
5188 "type": "object",
5189 "allOf": [
5190     { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5191     { "$ref": "#/definitions/oic.rd.attributes" }
5192 ],
5193 "required": ["sel"]
5194 }
5195

```

```

5196         example: /
5197             {
5198                 "rt": ["oic.wk.rd"],
5199                 "sel": 50
5200             }
5201
5202     post:
5203         description: |
5204             Publish the resource information
5205             Appropriates parts of the information posted will be discovered through /oic/res
5206
5207     queryParameters:
5208         rt:
5209             enum: oic.wk.rdpub
5210             type: string
5211         description: Only one Resource Type is used for GET; RT is optional
5212     Example: GET /oic/rd?rt=oic.wk.rdpub
5213
5214         required: false
5215     is : ['rddefinterface']
5216     body:
5217         application/json:
5218             schema: /
5219                 {
5220                     "$schema": "http://json-schema.org/draft-04/schema#",
5221                     "description" : "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5222 reserved.",
5223                     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
5224 schema.json#",
5225                     "title": "RD Publish & Update",
5226                     "definitions": {
5227                         "oic.rd.publish": {
5228                             "description": "Publishes resources as OIC Links into the resource directory",
5229                             "properties": {
5230                                 "linkSet": {
5231                                     "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-tagged-
5232 setoflinks"
5233                                 },
5234                                 "ttl": {
5235                                     "type": "integer",
5236                                     "description": "Time to indicate a RD, how long to keep this published item.
5237 After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
5238 publishing device updates the ttl using the update schema"
5239                                 }
5240                             }
5241                         }
5242                     },
5243                     "type": "object",
5244                     "allOf": [
5245                         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5246                         { "$ref": "#/definitions/oic.rd.publish" }
5247                     ],
5248                     "required": [ "links" ],
5249                     "dependencies": {
5250                         "links": [ "ttl" ]
5251                     }
5252                 }
5253
5254     responses :
5255         200:
5256             description: |
5257                 Respond with the same schema as publish but with the links have the "ins" parameter set
5258 to the appropriate instance value.

```

5259 This value is used by the receiver to manage that OIC Link instance.

5260

5261 body:

5262 application/json:

5263 schema: /

```
5264 {
5265     "$schema": "http://json-schema.org/draft-04/schema#",
5266     "description": "Copyright (c) 2016 Open Connectivity Foundation, Inc. All rights
5267 reserved.",
5268     "id": "http://www.openconnectivity.org/ocf-apis/core/schemas/oic.rd.publish-
5269 schema.json#",
5270     "title": "RD Publish & Update",
5271     "definitions": {
5272         "oic.rd.publish": {
5273             "description": "Publishes resources as OIC Links into the resource directory",
5274             "properties": {
5275                 "linkSet": {
5276                     "$ref": "oic.collection-schema.json#/definitions/oic.collection.setof-
5277 tagged-setoflinks"
5278                 },
5279                 "ttl": {
5280                     "type": "integer",
5281                     "description": "Time to indicate a RD, how long to keep this published
5282 item. After this time (in seconds) elapses, the RD invalidates the links. To keep link alive the
5283 publishing device updates the ttl using the update schema"
5284                 }
5285             }
5286         }
5287     },
5288     "type": "object",
5289     "allOf": [
5290         { "$ref": "oic.core-schema.json#/definitions/oic.core" },
5291         { "$ref": "#/definitions/oic.rd.publish" }
5292     ],
5293     "required": [ "links" ],
5294     "dependencies": {
5295         "links": [ "ttl" ]
5296     }
5297 }
5298
```

5299 example: /

```
5300 {
5301     "links": [
5302         {
5303             "href": "coap://someAuthority:1000/somePath",
5304             "rt": [ "oic.r.somerresource" ],
5305             "if": [ "oic.if.a" ],
5306             "ins": 12345,
5307             "eps": [
5308                 { "ep": "coap://[fe80::b1d6]:1111", "pri": 2 },
5309                 { "ep": "coaps://[fe80::b1d6]:1122" },
5310                 { "ep": "coap+tcp://[2001:db8:a::123]:2222", "pri": 3 }
5311             ]
5312         },
5313         {
5314             "href": "coap://someAuthority:1000/somePath",
5315             "rt": [ "oic.r.someotherresource" ],
5316             "if": [ "oic.if.baseline" ],
5317             "ins": 54321,
5318             "eps": [
5319                 { "ep": "coap://[[2001:db8:a::123]:2222" }
5320             ]
5321         }
5322     ],
5323     "ttl": 600
5324 }
5325
```

5326 delete:

```

5327     description: |
5328         Delete a particular OIC Link - the link may be a simple link or a link in a tagged set.
5329
5330     queryParameters:
5331         di:
5332             type: string
5333
5334             description: This is used to determine which set of links to operata on. (Need
5335 authentication to ensure that there is no spoofing). If instance is ommitted then the entire set of
5336 links from this device ID is deleted
5337
5338             required: true
5339
5340             example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1"
5341
5342         ins:
5343             type: string
5344
5345             description: Instance of the link to delete
5346 Value of parameter is a string where instance to be deleted are comma separated
5347
5348             required: false
5349
5350             example: DELETE /oic/rd?di="0685B960-736F-46F7-BEC0-9E6CBD671ADC1";ins="20"
5351
5352     responses :
5353         200:
5354             description: |
5355                 The delete succeeded

```

### 5353 D.13.5 Property Definition

| Property name | Value type                    | Mandatory | Access mode | Description   |
|---------------|-------------------------------|-----------|-------------|---|
| sel           | multiple types:<br>see schema | yes       | Read Write  | Selection criteria that a device wanting to publish to any RD can use to choose this Resource Directory over others that are discovered |
| di            | multiple types:<br>see schema |           | Read Write  | A unique identifier for the Resource Directory - the same as the device ID of the RD  |

### 5354 D.13.6 CRUDN behavior

| Resource | Create | Read | Update | Delete | Notify |
|----------|--------|------|--------|--------|--------|
| /oic/rd  |        | get  | post   | delete |        |

5355