



**OCF “Dubai” – Remove OIC 1.1 Security Features – Security WG CR 2719**

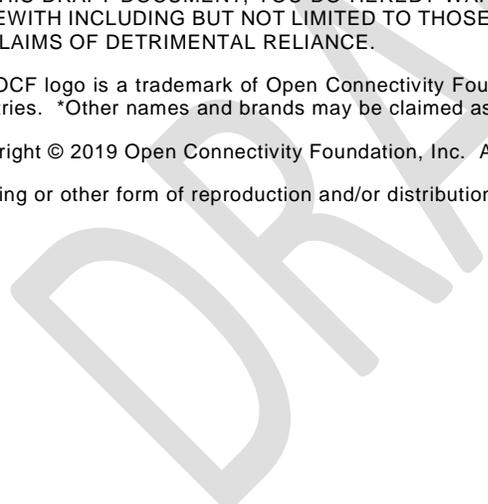
Legal Disclaimer

THIS IS A DRAFT SPECIFICATION DOCUMENT ONLY AND HAS NOT BEEN ADOPTED BY THE OPEN CONNECTIVITY FOUNDATION. THIS DRAFT DOCUMENT MAY NOT BE RELIED UPON FOR ANY PURPOSE OTHER THAN REVIEW OF THE CURRENT STATE OF THE DEVELOPMENT OF THIS DRAFT DOCUMENT. THE OPEN CONNECTIVITY FOUNDATION AND ITS MEMBERS RESERVE THE RIGHT WITHOUT NOTICE TO YOU TO CHANGE ANY OR ALL PORTIONS HEREOF, DELETE PORTIONS HEREOF, MAKE ADDITIONS HERETO, DISCARD THIS DRAFT DOCUMENT IN ITS ENTIRETY OR OTHERWISE MODIFY THIS DRAFT DOCUMENT AT ANY TIME. YOU SHOULD NOT AND MAY NOT RELY UPON THIS DRAFT DOCUMENT IN ANY WAY, INCLUDING BUT NOT LIMITED TO THE DEVELOPMENT OF ANY PRODUCTS OR SERVICES. IMPLEMENTATION OF THIS DRAFT DOCUMENT IS DONE AT YOUR OWN RISK AMEND AND IT IS NOT SUBJECT TO ANY LICENSING GRANTS OR COMMITMENTS UNDER THE OPEN CONNECTIVITY FOUNDATION INTELLECTUAL PROPERTY RIGHTS POLICY OR OTHERWISE. IN CONSIDERATION OF THE OPEN CONNECTIVITY FOUNDATION GRANTING YOU ACCESS TO THIS DRAFT DOCUMENT, YOU DO HEREBY WAIVE ANY AND ALL CLAIMS ASSOCIATED HERewith INCLUDING BUT NOT LIMITED TO THOSE CLAIMS DISCUSSED BELOW, AS WELL AS CLAIMS OF DETRIMENTAL RELIANCE.

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

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



28 **NOTE: This CR incorporates the removal of OIC 1.1 features, as OCF no longer**  
29 **supports backwards compatibility with OIC 1.1 devices.**

30 1 Scope .....1  
31 2 Normative References .....1  
32 3 Terms, definitions, and abbreviated terms .....3  
33 3.1 Terms and definitions .....3  
34 3.2 Abbreviated terms .....6  
35 4 Document Conventions and Organization.....9  
36 4.1 Conventions .....9  
37 4.2 Notation .....10  
38 4.3 Data types .....11  
39 4.4 Document structure .....11  
40 5 Security Overview.....12  
41 5.1 Preamble .....12  
42 5.2 Access Control.....14  
43 5.2.1 ACL Architecture .....16  
44 5.2.2 Access Control Scoping Levels .....19  
45 5.3 Onboarding Overview .....21  
46 5.3.1 Onboarding General .....21  
47 5.3.2 OnBoarding Steps .....23  
48 5.3.3 Establishing a Device Owner.....24  
49 5.3.4 Provisioning for Normal Operation.....25  
50 5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview .....25  
51 5.3.6 OCF Compliance Management System.....25  
52 5.4 Provisioning .....26  
53 5.4.1 Provisioning General .....26  
54 5.4.2 Provisioning other services .....26  
55 5.4.3 Provisioning Credentials for Normal Operation.....26  
56 5.4.4 Role Assignment and Provisioning for Normal Operation .....27  
57 5.4.5 ACL provisioning .....27  
58 5.5 Secure Resource Manager (SRM) .....27  
59 5.6 Credential Overview .....28  
60 6 Security for the Discovery Process .....29  
61 6.1 Preamble .....29  
62 6.2 Security Considerations for Discovery .....29  
63 7 Security Provisioning .....32  
64 7.1 Device Identity .....32  
65 7.1.1 General Device Identity .....32  
66 7.1.2 Device Identity for Devices with UAID.....32  
67 7.2 Device Ownership .....34  
68 7.3 Device Ownership Transfer Methods .....34  
69 7.3.1 OTM implementation requirements .....34  
70 7.3.2 SharedKey Credential Calculation .....36

71	7.3.3	Certificate Credential Generation .....	36
72	7.3.4	Just-Works OTM .....	36
73	7.3.5	Random PIN Based OTM.....	38
74	7.3.6	Manufacturer Certificate Based OTM.....	41
75	7.3.7	Vendor Specific OTMs.....	43
76	7.3.8	Establishing Owner Credentials.....	44
77	7.3.9	Security considerations regarding selecting an Ownership Transfer Method ..	53
78	7.3.10	Security Profile Assignment .....	53
79	7.4	Provisioning .....	54
80	7.4.1	Provisioning Flows .....	54
81	7.5	Device Provisioning for OCF Cloud .....	59
82	7.5.1	Cloud Provisioning General .....	59
83	7.5.2	Device Provisioning by Mediator .....	59
84	8	Device Onboarding State Definitions.....	60
85	8.1	Device Onboarding General .....	60
86	8.2	Device Onboarding-Reset State Definition .....	62
87	8.3	Device Ready-for-OTM State Definition .....	62
88	8.4	Device Ready-for-Provisioning State Definition .....	63
89	8.5	Device Ready-for-Normal-Operation State Definition .....	63
90	8.6	Device Soft Reset State Definition.....	64
91	9	Security Credential Management .....	66
92	9.1	Preamble .....	66
93	9.2	Credential Lifecycle.....	66
94	9.2.1	Credential Lifecycle General .....	66
95	9.2.2	Creation.....	66
96	9.2.3	Deletion .....	66
97	9.2.4	Refresh.....	66
98	9.2.5	Revocation.....	66
99	9.3	Credential Types .....	67
100	9.3.1	Preamble .....	67
101	9.3.2	Pair-wise Symmetric Key Credentials .....	67
102	9.3.3	Group Symmetric Key Credentials.....	67
103	9.3.4	Asymmetric Authentication Key Credentials .....	68
104	9.3.5	Asymmetric Key Encryption Key Credentials.....	68
105	9.3.6	Certificate Credentials .....	69
106	9.3.7	Password Credentials.....	69
107	9.4	Certificate Based Key Management.....	69
108	9.4.1	Overview .....	69
109	9.4.2	X.509 Digital Certificate Profiles.....	70
110	9.4.3	Certificate Revocation List (CRL) Profile.....	79
111	9.4.4	Resource Model .....	80
112	9.4.5	Certificate Provisioning.....	80
113	9.4.6	CRL Provisioning .....	81
114	10	Device Authentication .....	83

115	10.1	Device Authentication General .....	83
116	10.2	Device Authentication with Symmetric Key Credentials.....	83
117	10.3	Device Authentication with Raw Asymmetric Key Credentials .....	83
118	10.4	Device Authentication with Certificates .....	83
119	10.4.1	Device Authentication with Certificates General .....	83
120	10.4.2	Role Assertion with Certificates.....	84
121	10.4.3	OCF PKI Roots .....	85
122	10.4.4	PKI Trust Store .....	85
123	10.4.5	Path Validation and extension processing.....	86
124	10.5	Device Authentication with OCF Cloud .....	87
125	10.5.1	Device Authentication with OCF Cloud General.....	87
126	10.5.2	Device Connection with the OCF Cloud .....	87
127	10.5.3	Security Considerations.....	88
128	11	Message Integrity and Confidentiality .....	90
129	11.1	Preamble .....	90
130	11.2	Session Protection with DTLS .....	90
131	11.2.1	DTLS Protection General .....	90
132	11.2.2	Unicast Session Semantics.....	90
133	11.2.3	Cloud Session Semantics .....	90
134	11.3	Cipher Suites .....	90
135	11.3.1	Cipher Suites General .....	90
136	11.3.2	Cipher Suites for Device Ownership Transfer.....	90
137	11.3.3	Cipher Suites for Symmetric Keys .....	91
138	11.3.4	Cipher Suites for Asymmetric Credentials .....	92
139	11.3.5	Cipher suites for OCF Cloud Credentials .....	92
140	12	Access Control .....	94
141	12.1	ACL Generation and Management.....	94
142	12.2	ACL Evaluation and Enforcement.....	94
143	12.2.1	ACL Evaluation and Enforcement General .....	94
144	12.2.2	Host Reference Matching.....	94
145	12.2.3	Resource Wildcard Matching.....	94
146	12.2.4	Multiple Criteria Matching .....	95
147	12.2.5	Subject Matching using Wildcards .....	95
148	12.2.6	Subject Matching using Roles .....	95
149	12.2.7	ACL Evaluation .....	96
150	13	Security Resources.....	98
151	13.1	Security Resources General .....	98
152	13.2	Device Owner Transfer Resource.....	100
153	13.2.1	Device Owner Transfer Resource General .....	100
154	13.2.2	Persistent and Semi-persistent Device Identifiers.....	103
155	13.2.3	Onboarding Considerations for Device Identifier .....	103
156	13.2.4	OCF defined OTMs.....	104
157	13.3	Credential Resource .....	105
158	13.3.1	Credential Resource General .....	105

159	13.3.2	Properties of the Credential Resource .....	109
160	13.3.3	Key Formatting.....	112
161	13.3.4	Credential Refresh Method Details .....	112
162	13.4	Certificate Revocation List .....	114
163	13.4.1	CRL Resource Definition .....	114
164	13.5	ACL Resources .....	114
165	13.5.1	ACL Resources General .....	114
166	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures.....	114
167	13.5.3	ACL Resource.....	115
168	13.6	Access Manager ACL Resource .....	124
169	13.7	Signed ACL Resource .....	125
170	13.8	Provisioning Status Resource .....	126
171	13.9	Certificate Signing Request Resource .....	131
172	13.10	Roles Resource.....	132
173	13.11	Account Resource .....	133
174	13.12	Account Session resource .....	135
175	13.13	Account Token Refresh Resource .....	136
176	13.14	Security Virtual Resources (SVRs) and Access Policy .....	137
177	13.15	SVRs, Discoverability and OCF Endpoints .....	137
178	13.16	Additional Privacy Consideration for Core and SVRs Resources .....	137
179	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ....	137
180	13.16.2	Privacy Protecting the Device Identifiers.....	140
181	13.16.3	Privacy Protecting the Protocol Independent Device Identifier.....	140
182	13.16.4	Privacy Protecting the Platform Identifier .....	141
183	13.17	Easy Setup Resource Device State .....	141
184	14	Security Hardening Guidelines/ Execution Environment Security.....	144
185	14.1	Preamble .....	144
186	14.2	Execution Environment Elements .....	144
187	14.2.1	Execution Environment Elements General .....	144
188	14.2.2	Secure Storage .....	144
189	14.2.3	Secure execution engine .....	147
190	14.2.4	Trusted input/output paths .....	147
191	14.2.5	Secure clock .....	147
192	14.2.6	Approved algorithms.....	147
193	14.2.7	Hardware tamper protection.....	148
194	14.3	Secure Boot.....	148
195	14.3.1	Concept of software module authentication.....	148
196	14.3.2	Secure Boot process .....	150
197	14.3.3	Robustness Requirements .....	150
198	14.4	Attestation .....	150
199	14.5	Software Update .....	150
200	14.5.1	Overview: .....	150
201	14.5.2	Recognition of Current Differences.....	151
202	14.5.3	Software Version Validation .....	151

203	14.5.4	Software Update .....	151
204	14.5.5	Recommended Usage .....	151
205	14.6	Non-OCF Endpoint interoperability .....	152
206	14.7	Security Levels.....	152
207	14.8	Security Profiles .....	152
208	14.8.1	Security Profiles General .....	152
209	14.8.2	Identification of Security Profiles (Normative).....	153
210	14.8.3	Security Profiles .....	154
211	15	Device Type Specific Requirements.....	160
212	15.1	Bridging Security.....	160
213	15.1.1	Universal Requirements for Bridging to another Ecosystem .....	160
214	15.1.2	Additional Security Requirements specific to Bridged Protocols .....	160
215	Annex A (informative)	Access Control Examples .....	162
216	A.1	Example OCF ACL Resource .....	162
217	A.2	Example AMS .....	162
218	Annex B (Informative)	Execution Environment Security Profiles.....	164
219	Annex C (normative)	Resource Type definitions .....	165
220	C.1	List of Resource Type definitions .....	165
221	C.2	Account Token .....	165
222	C.2.1	Introduction.....	165
223	C.2.2	Well-known URI .....	165
224	C.2.3	Resource type.....	165
225	C.2.4	OpenAPI 2.0 definition.....	165
226	C.2.5	Property definition .....	168
227	C.2.6	CRUDN behaviour .....	169
228	C.3	Access Control List.....	169
229	C.3.1	Introduction.....	169
230	C.3.2	Well-known URI .....	169
231	C.3.3	Resource type.....	169
232	C.3.4	OpenAPI 2.0 definition.....	169
233	C.3.5	Property definition .....	176
234	C.3.6	CRUDN behaviour .....	176
235	C.4	Access Control List-2.....	177
236	C.4.1	Introduction.....	177
237	C.4.2	Well-known URI .....	177
238	C.4.3	Resource type.....	177
239	C.4.4	OpenAPI 2.0 definition.....	177
240	C.4.5	Property definition .....	186
241	C.4.6	CRUDN behaviour .....	187
242	C.5	Managed Access Control .....	187
243	C.5.1	Introduction.....	187
244	C.5.2	Well-known URI .....	187
245	C.5.3	Resource type.....	187
246	C.5.4	OpenAPI 2.0 definition.....	187

247	C.5.5	Property definition .....	190
248	C.5.6	CRUDN behaviour .....	191
249	C.6	Credential .....	191
250	C.6.1	Introduction .....	191
251	C.6.2	Well-known URI .....	191
252	C.6.3	Resource type .....	191
253	C.6.4	OpenAPI 2.0 definition .....	191
254	C.6.5	Property definition .....	201
255	C.6.6	CRUDN behaviour .....	202
256	C.7	Certificate Revocation .....	202
257	C.7.1	Introduction .....	202
258	C.7.2	Well-known URI .....	202
259	C.7.3	Resource type .....	202
260	C.7.4	OpenAPI 2.0 definition .....	202
261	C.7.5	Property definition .....	204
262	C.7.6	CRUDN behaviour .....	205
263	C.8	Certificate Signing Request .....	205
264	C.8.1	Introduction .....	205
265	C.8.2	Well-known URI .....	205
266	C.8.3	Resource type .....	205
267	C.8.4	OpenAPI 2.0 definition .....	205
268	C.8.5	Property definition .....	207
269	C.8.6	CRUDN behaviour .....	207
270	C.9	Device Owner Transfer Method .....	208
271	C.9.1	Introduction .....	208
272	C.9.2	Well-known URI .....	208
273	C.9.3	Resource type .....	208
274	C.9.4	OpenAPI 2.0 definition .....	208
275	C.9.5	Property definition .....	211
276	C.9.6	CRUDN behaviour .....	213
277	C.10	Device Provisioning Status .....	213
278	C.10.1	Introduction .....	213
279	C.10.2	Well-known URI .....	213
280	C.10.3	Resource type .....	213
281	C.10.4	OpenAPI 2.0 definition .....	213
282	C.10.5	Property definition .....	217
283	C.10.6	CRUDN behaviour .....	222
284	C.11	Asserted Roles .....	222
285	C.11.1	Introduction .....	222
286	C.11.2	Well-known URI .....	222
287	C.11.3	Resource type .....	222
288	C.11.4	OpenAPI 2.0 definition .....	222
289	C.11.5	Property definition .....	231
290	C.11.6	CRUDN behaviour .....	231

291	C.12 Signed Access Control List .....	232
292	C.12.1 Introduction .....	232
293	C.12.2 Well-known URI .....	232
294	C.12.3 Resource type .....	232
295	C.12.4 OpenAPI 2.0 definition .....	232
296	C.12.5 Property definition .....	239
297	C.12.6 CRUDN behaviour .....	240
298	C.13 Session .....	240
299	C.13.1 Introduction .....	240
300	C.13.2 Well-known URI .....	240
301	C.13.3 Resource type .....	240
302	C.13.4 OpenAPI 2.0 definition .....	240
303	C.13.5 Property definition .....	242
304	C.13.6 CRUDN behaviour .....	243
305	C.14 Security Profile .....	243
306	C.14.1 Introduction .....	243
307	C.14.2 Well-known URI .....	243
308	C.14.3 Resource type .....	243
309	C.14.4 OpenAPI 2.0 definition .....	243
310	C.14.5 Property definition .....	245
311	C.14.6 CRUDN behaviour .....	246
312	C.15 Token Refresh .....	246
313	C.15.1 Introduction .....	246
314	C.15.2 Well-known URI .....	246
315	C.15.3 Resource type .....	246
316	C.15.4 OpenAPI 2.0 definition .....	246
317	C.15.5 Property definition .....	248
318	C.15.6 CRUDN behaviour .....	249
319	Annex D (informative) OID definitions .....	250
320	Annex E (informative) Security considerations specific to Bridged Protocols .....	252
321	E.1 Security Considerations specific to the AllJoyn Protocol .....	252
322	E.2 Security Considerations specific to the Bluetooth LE Protocol .....	252
323	E.3 Security Considerations specific to the oneM2M Protocol .....	252
324	E.4 Security Considerations specific to the U+ Protocol .....	252
325	E.5 Security Considerations specific to the Z-Wave Protocol .....	253
326	E.6 Security Considerations specific to the Zigbee Protocol .....	254
327		

328 **FIGURES**

329 Figure 1 – OCF Interaction ..... 10

330 Figure 2 – OCF Layers..... 12

331 Figure 3 – OCF Security Enforcement Points ..... 14

332 Figure 4 – Use case-1 showing simple ACL enforcement ..... 17

333 Figure 5 – Use case 2: A policy for the requested Resource is missing..... 17

334 Figure 6 – Use case-3 showing AMS supported ACL ..... 18

335 Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS..... 19

336 Figure 8 – Example Resource definition with opaque Properties.....20

337 Figure 9 – Property Level Access Control .....20

338 Figure 10 – Onboarding Overview .....22

339 Figure 11 – OCF Onboarding Process .....24

340 Figure 12 – OCF's SRM Architecture .....28

341 Figure 13 – Discover New Device Sequence .....35

342 Figure 14 – A Just Works OTM .....37

343 Figure 15 – Random PIN-based OTM .....39

344 Figure 16 – Manufacturer Certificate Based OTM Sequence .....42

345 Figure 17 – Vendor-specific Owner Transfer Sequence.....44

346 Figure 18 – Establish Device Identity Flow.....45

347 Figure 19 – Owner Credential Selection Provisioning Sequence.....46

348 Figure 20 – Symmetric Owner Credential Provisioning Sequence .....47

349 Figure 21 – Asymmetric Owner Credential Provisioning Sequence .....49

350 Figure 22 – Configure Device Services .....51

351 Figure 23 – Provision New Device for Peer to Peer Interaction Sequence.....52

352 Figure 24 – Example of Client-directed provisioning .....55

353 Figure 25 – Example of Server-directed provisioning using a single provisioning service .....56

354 Figure 26 – Example of Server-directed provisioning involving multiple support services .....59

355 Figure 27 – Device state model .....61

356 Figure 28 – OBT Sanity Check Sequence in SRESET .....64

357 Figure 29 – Client-directed Certificate Transfer .....81

358 Figure 30 – Client-directed CRL Transfer .....82

359 Figure 32 – Asserting a role with a certificate role credential.....85

360 Figure 33 – Device connection with OCF Cloud.....88

361 Figure 34 – OCF Security Resources.....98

362 Figure 35 – /oic/sec/cred Resource and Properties.....99

363 Figure 36 – /oic/sec/acl2 Resource and Properties .....99

364 Figure 37 – /oic/sec/amacl Resource and Properties .....100

365 Figure 38 – /oic/sec/sacl Resource and Properties .....100

366 Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states ..... 141

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

367 Figure 40 – Software Module Authentication ..... 149  
368 Figure 41 – Verification Software Module ..... 149  
369 Figure 42 – Software Module Authenticity ..... 150  
370 Figure A-1 – Example /oic/sec/acl2 Resource..... 162  
371 Figure A-2 Example /oic/sec/amacl Resource..... 163  
372 Figure E-1 Security Considerations for BLE Bridge..... 252  
373 Figure E-2 Security Considerations for Z-Wave Bridge ..... 254  
374 Figure E-3 Security Considerations for Zigbee Bridge ..... 255  
375

376	<b>Tables</b>	
377	Table 1 – Discover New Device Details.....	35
378	Table 2 – A Just Works OTM Details.....	37
379	Table 3 – Random PIN-based OTM Details .....	39
380	Table 4 – Manufacturer Certificate Based OTM Details .....	43
381	Table 5 – Vendor-specific Owner Transfer Details .....	44
382	Table 6 – Establish Device Identity Details.....	46
383	Table 7 – Owner Credential Selection Details.....	47
384	Table 8 – Symmetric Owner Credential Assignment Details .....	47
385	Table 9 – Asymmetric Owner Credential Assignment Details.....	49
386	Table 10 – Configure Device Services Detail.....	51
387	Table 11 – Provision New Device for Peer to Peer Details .....	52
388	Table 12 – Steps describing Client -directed provisioning .....	55
389	Table 13 – Steps for Server-directed provisioning using a single provisioning service .....	56
390	Table 14 – Steps for Server-directed provisioning involving multiple support services .....	59
391	Table 15 – Mapping of Properties of the oic.r.account and oic.r.coapcloudconf Resources ...	60
392	Table 16 – X.509 v1 fields for Root CA Certificates .....	70
393	Table 17 - X.509 v3 extensions for Root CA Certificates .....	71
394	Table 18 - X.509 v1 fields for Intermediate CA Certificates .....	71
395	Table 19 – X.509 v3 extensions for Intermediate CA Certificates.....	72
396	Table 20 – X.509 v1 fields for End-Entity Certificates .....	72
397	Table 21 – X.509 v3 extensions for End-Entity Certificates .....	73
398	Table 22 – Device connection with the OCF Cloud flow .....	88
399	Table 23 – ACE2 Wildcard Matching Strings Description.....	94
400	Table 24 – Definition of the /oic/sec/doxm Resource.....	101
401	Table 25 – Properties of the /oic/sec/doxm Resource .....	101
402	Table 26 – Properties of the /oic/sec/didtype Property .....	102
403	Table 27 – Properties of the oic.sec.doxmtype Property.....	104
404	Table 28 – Definition of the oic.r.cred Resource .....	105
405	Table 29 – Properties of the /oic/sec/cred Resource .....	106
406	Table 30 – Properties of the oic.sec.cred Property.....	107
407	Table 31: Properties of the oic.sec.credusagetype Property .....	108
408	Table 32 – Properties of the oic.sec.pubdatatype Property.....	108
409	Table 33 – Properties of the oic.sec.privdatatype Property.....	108
410	Table 34 – Properties of the oic.sec.optdatatype Property.....	109
411	Table 35 – Definition of the oic.sec.roletype Property. ....	109
412	Table 36 – Value Definition of the oic.sec.crmttype Property .....	111
413	Table 37 – 128-bit symmetric key .....	112
414	Table 38 – 256-bit symmetric key .....	112

415	Table 39 – Definition of the oic.r.crl Resource .....	114
416	Table 40 – Properties of the oic.r.crl Resource .....	114
417	Table 41 – BNF Definition of OCF ACL .....	114
418	Table 42 – Definition of the oic.r.acl Resource .....	116
419	Table 43 – Properties of the oic.r.acl Resource .....	117
420	Table 44 – Properties of the oic.r.ace Property .....	119
421	Table 45 – Value Definition of the oic.sec.crudntype Property .....	120
422	Table 46 – Definition of the oic.sec.acl2 Resource .....	120
423	Table 47 – Properties of the oic.sec.acl2 Resource .....	121
424	Table 48 – oic.sec.ace2 data type definition .....	122
425	Table 49 – oic.sec.ace2.resource-ref data type definition .....	122
426	Table 50 – Value definition oic.sec.conntype Property .....	122
427	Table 51 – Definition of the oic.r.amacl Resource .....	124
428	Table 52 – Properties of the oic.r.amacl Resource .....	125
429	Table 53 – Definition of the oic.r.sacl Resource .....	125
430	Table 54 – Properties of the oic.r.sacl Resource .....	125
431	Table 55 – Properties of the oic.sec.sigtype Property .....	126
432	Table 56 – Definition of the oic.r.pstat Resource .....	126
433	Table 57 – Properties of the oic.r.pstat Resource .....	127
434	Table 58 – Properties of the /oic/sec/dostype Property .....	128
435	Table 59 – Definition of the oic.sec.dpmtype Property .....	130
436	Table 60 – Value Definition of the oic.sec.dpmtype Property (Low-Byte) .....	130
437	Table 61 – Value Definition of the oic.sec.dpmtype Property (High-Byte) .....	130
438	Table 62 – Definition of the oic.sec.pomtype Property .....	131
439	Table 63 – Value Definition of the oic.sec.pomtype Property .....	131
440	Table 64 – Definition of the oic.r.csr Resource .....	131
441	Table 65 – Properties of the oic.r.csr Resource .....	132
442	Table 66 – Definition of the oic.r.roles Resource .....	133
443	Table 67 – Properties of the oic.r.roles Resource .....	133
444	Table 68 – Definition of the oic.r.account Resource .....	134
445	Table 69 – Properties of the oic.r.account Resource .....	135
446	Table 70 – Definition of the oic.r.session Resource .....	135
447	Table 71 – Properties of the oic.r.session Resource .....	136
448	Table 72 – Definition of the oic.r.tokenrefresh Resource .....	137
449	Table 73 – Properties of the oic.r.tokenrefresh Resource .....	137
450	Table 74 – Core Resource Properties Access Modes given various Device States .....	139
451	Table 75 – Examples of Sensitive Data .....	145
452	Table 76 – Definition of the oic.sec.sp Resource .....	154
453	Table 77 – Properties of the oic.sec.sp Resource .....	154

454	Table B.1 – OCF Security Profile .....	164
455	Table C.1 – Alphabetized list of security resources.....	165
456	Table C.2 – The Property definitions of the Resource with type 'rt' = ['oic.r.account'] .....	168
457	Table C.3 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.account'] .....	169
458	Table C.4 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl'] .....	176
459	Table C.5 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl'] .....	176
460	Table C.6 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl2'] .....	186
461	Table C.7 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl2'] .....	187
462	Table C.8 – The Property definitions of the Resource with type 'rt' = ['oic.r.amacl'] .....	190
463	Table C.9 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.amacl'] .....	191
464	Table C.10 – The Property definitions of the Resource with type 'rt' = ['oic.r.cred'] .....	201
465	Table C.11 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.cred'] .....	202
466	Table C.12 – The Property definitions of the Resource with type 'rt' = ['oic.r.crl'] .....	204
467	Table C.13 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.crl'] .....	205
468	Table C.14 – The Property definitions of the Resource with type 'rt' = ['oic.r.csr'] .....	207
469	Table C.15 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.csr'] .....	207
470	Table C.16 – The Property definitions of the Resource with type 'rt' = ['oic.r.doxm'] .....	211
471	Table C.17 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.doxm'] .....	213
472	Table C.18 – The Property definitions of the Resource with type 'rt' = ['oic.r.pstat'] .....	217
473	Table C.19 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.pstat'] .....	222
474	Table C.20 – The Property definitions of the Resource with type 'rt' = ['oic.r.roles'] .....	231
475	Table C.21 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.roles'] .....	231
476	Table C.22 – The Property definitions of the Resource with type 'rt' = ['oic.r.sacl'] .....	239
477	Table C.23 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sacl'] .....	240
478	Table C.24 – The Property definitions of the Resource with type 'rt' = ['oic.r.session'] .....	242
479	Table C.25 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.session'] .....	243
480	Table C.26 – The Property definitions of the Resource with type 'rt' = ['oic.r.sp'] .....	245
481	Table C.27 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sp'] .....	246
482	Table C.28 – The Property definitions of the Resource with type 'rt' = ['oic.r.tokenrefresh'] ..	248
483	Table C.29 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.tokenrefresh'] ..	249
484	Table E.1 GAP security mode .....	252
485	Table E.2 TLS 1.2 Cipher Suites used by U+ .....	253
486	Table E.3 Z-Wave Security Class .....	254
487	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers.....	255
488		
489		

490 **1 Scope**

491 This document defines security objectives, philosophy, resources and mechanism that impacts  
492 OCF base layers of ISO/IEC 30118-1:2018. ISO/IEC 30118-1:2018 contains informative security  
493 content. The OCF Security Specification contains security normative content and may contain  
494 informative content related to the OCF base or other OCF documents.

495 **2 Normative References**

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

499 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)  
500 Specification -- Part 1: Core specification  
501 <https://www.iso.org/standard/53238.html>  
502 Latest version available at:  
503 [https://openconnectivity.org/specs/OCF\\_Core\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Core_Specification.pdf)

504 ISO/IEC 30118-3:2018 Information technology -- Open Connectivity Foundation (OCF)  
505 Specification -- Part 3: Bridging specification  
506 <https://www.iso.org/standard/74240.html>  
507 Latest version available at:  
508 [https://openconnectivity.org/specs/OCF\\_Bridging\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf)

509 ISO/IEC 30118-4:2018 Information technology -- Open Connectivity Foundation (OCF)  
510 Specification -- Part 4: Resource type specification  
511 <https://www.iso.org/standard/74241.html>  
512 Latest version available at:  
513 [https://openconnectivity.org/specs/OCF\\_Resource\\_to\\_AllJoyn\\_Interface\\_Mapping.pdf](https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping.pdf)

514 OCF Wi-Fi Easy Setup, *Open Connectivity Foundation Wi-Fi Easy Setup*, Version 2.0.1  
515 Latest version available at:  
516 [https://openconnectivity.org/specs/OCF\\_Wi-Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)

517 OCF Cloud Specification, *Open Connectivity Foundation Cloud*, Version 2.0.1  
518 Latest version available at:  
519 [https://openconnectivity.org/specs/OCF\\_Cloud\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf)

520 JSON SCHEMA, draft version 4, <http://json-schema.org/latest/json-schema-core.html>.

521 IETF RFC 2315, *PKCS #7: Cryptographic Message Syntax Version 1.5*, March 1998,  
522 <https://tools.ietf.org/html/rfc2315>

523 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September  
524 2000, <https://tools.ietf.org/html/rfc2898>

525 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November  
526 2000, <https://tools.ietf.org/html/rfc2986>

527 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December  
528 2005, <https://tools.ietf.org/html/rfc4279>

529 IETF RFC 4492, *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security  
530 (TLS)*, May 2006, <https://tools.ietf.org/html/rfc4492>

531 IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, August 2008,  
532 <https://tools.ietf.org/html/rfc5246>

533 IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation*  
534 *List (CRL) Profile*, May 2008, <https://tools.ietf.org/html/rfc5280>

535 IETF RFC 5489, *ECDHE\_PSK Cipher Suites for Transport Layer Security (TLS)*, March 2009,  
536 <https://tools.ietf.org/html/rfc5489>

537 IETF RFC 5545, *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*,  
538 September 2009, <https://tools.ietf.org/html/rfc5545>

539 IETF RFC 5755, *An Internet Attribute Certificate Profile for Authorization*, January 2010,  
540 <https://tools.ietf.org/html/rfc5755>

541 IETF RFC 6347, *Datagram Transport Layer Security Version 1.2*, January 2012,  
542 <https://tools.ietf.org/html/rfc6347>

543 IETF RFC 6655, *AES-CCM Cipher Suites for Transport Layer Security (TLS)*, July 2012,  
544 <https://tools.ietf.org/html/rfc6655>

545 IETF RFC 6749, *The OAuth 2.0 Authorization Framework*, October 2012,  
546 <https://tools.ietf.org/html/rfc6749>

547 IETF RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*, October 2012,  
548 <https://tools.ietf.org/html/rfc6750>

549 IETF RFC 7228, *Terminology for Constrained-Node Networks*, May 2014,  
550 <https://tools.ietf.org/html/rfc7228>

551 IETF RFC 7250, *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram*  
552 *Transport Layer Security (DTLS)*, June 2014, <https://tools.ietf.org/html/rfc7250>

553 IETF RFC 7251, *AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS*, June 2014,  
554 <https://tools.ietf.org/html/rfc7251>

555 IETF RFC 7515, *JSON Web Signature (JWS)*, May 2015, <https://tools.ietf.org/html/rfc7515>

556 IETF RFC 7519, *JSON Web Token (JWT)*, May 2015, <https://tools.ietf.org/html/rfc7519>

557 IETF RFC 8323, *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*,  
558 February 2018, <https://tools.ietf.org/html/rfc8323>

559 IETF RFC 8392, *CBOR Web Token (CWT)*, May 2018, <https://tools.ietf.org/html/rfc8392>

560 oneM2M Release 3 Specifications, <http://www.onem2m.org/technical/published-drafts>

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

563

564 **3 Terms, definitions, and abbreviated terms**

565 **3.1 Terms and definitions**

566 For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2018 and  
567 the following apply.

568 ISO and IEC maintain terminological databases for use in standardization at the following  
569 addresses:

570 – ISO Online browsing platform: available at <https://www.iso.org/obp>

571 – IEC Electropedia: available at <http://www.electropedia.org/>

572 **3.1.1**

573 **Access Management Service (AMS)**

574 dynamically constructs ACL Resources in response to a Device Resource request.

575 Note 1 to entry: An AMS can evaluate access policies remotely and supply the result to a Server which allows or denies  
576 a pending access request. An AMS is authorised to provision ACL Resources.

577 **3.1.2**

578 **Access Token**

579 a credential used to access protected resources. An Access Token is a string representing an  
580 authorization issued to the client.

581 **3.1.3**

582 **Authorization Provider**

583 a Server issuing Access Tokens (3.1.2) to the Client after successfully authenticating the OCF  
584 Cloud User (3.1.16) and obtaining authorization.

585 Note 1 to entry: Also known as authorization server in IETF RFC 6749.

586 **3.1.4**

587 **Client**

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

589 **3.1.5**

590 **Credential Management Service (CMS)**

591 a name and Resource Type (oic.sec.cms) given to a Device that is authorized to provision  
592 credential Resources.

593 **3.1.6**

594 **Device**

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

596 **3.1.7**

597 **Device Class**

598 Note 1 to entry: As defined in IETF RFC 7228. IETF RFC 7228 defines classes of constrained devices that distinguish  
599 when the OCF small footprint stack is used vs. a large footprint stack. Class 2 and below is for small footprint stacks.

600 **3.1.8**

601 **Device ID**

602 a stack instance identifier.

603 **3.1.9**

604 **Device Ownership Transfer Service (DOTS)**

605 a logical entity that establishes device ownership

606 **3.1.10**  
607 **Device Registration**  
608 a process by which Device is enrolled/registered to the OCF Cloud infrastructure (using Device  
609 certificate and unique credential) and becomes ready for further remote operation through the cloud  
610 interface (e.g. connection to remote Resources or publishing of its own Resources for access).

611 **3.1.11**  
612 **End-Entity**  
613 any certificate holder which is not a Root or Intermediate Certificate Authority.

614 Note 1 to entry: Typically, a device certificate.

615 **3.1.12**  
616 **Entity**  
617 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

618 **3.1.13**  
619 **OCF Interface**  
620 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

621 **3.1.14**  
622 **Intermediary**  
623 a Device that implements both Client and Server roles and may perform protocol translation, virtual  
624 device to physical device mapping or Resource translation

625 **3.1.15**  
626 **OCF Cipher Suite**  
627 a set of algorithms and parameters that define the cryptographic functionality of a Device. The  
628 OCF Cipher Suite includes the definition of the public key group operations, signatures, and specific  
629 hashing and encoding used to support the public key.

630 **3.1.16**  
631 **OCF Cloud User**  
632 a person or organization authorizing a set of Devices to interact with each other via an OCF Cloud.

633 Note 1 to entry: For each of the Devices, the OCF Cloud User is either the same as, or a delegate of, the person or  
634 organization that onboarded that Device. The OCF Cloud User delegates, to the OCF Cloud authority, authority to route  
635 between Devices registered by the OCF Cloud User. The OCF Cloud delegates, to the OCF Cloud User, authority to  
636 select the set of Devices which can register and use the services of the OCF Cloud.

637 **3.1.17**  
638 **OCF Rooted Certificate Chain**  
639 a collection of X.509 v3 certificates in which each certificate chains to a trust anchor certificate  
640 which has been issued by a certificate authority under the direction, authority, and approval of the  
641 Open Connectivity Foundation Board of Directors as a trusted root for the OCF ecosystem.

642 **3.1.18**  
643 **Onboarding Tool (OBT)**  
644 a tool that implements DOTS(3.1.9), AMS(3.1.1) and CMS(3.1.5) functionality

645 **3.1.19**  
646 **Out of Band Method**  
647 any mechanism for delivery of a secret from one party to another, not specified by OCF

648 **3.1.20**  
649 **Owner Credential (OC)**  
650 Credential, provisioned by an Onboarding Tool to a Device during onboarding, for the purposes of  
651 mutual authentication of the Device and Onboarding Tool during subsequent interactions

652 **3.1.21**  
653 **Platform ID**  
654 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

655 **3.1.22**  
656 **Property**  
657 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

658 **3.1.23**  
659 **Resource**  
660 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

661 **3.1.24**  
662 **Role (Network context)**  
663 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

664 **3.1.25**  
665 **Role Identifier**  
666 a Property of an OCF credentials Resource or element in a role certificate that identifies a privileged  
667 role that a Server Device associates with a Client Device for the purposes of making authorization  
668 decisions when the Client Device requests access to Device Resources.

669 **3.1.26**  
670 **Secure Resource Manager (SRM)**  
671 a module in the OCF Core that implements security functionality that includes management of  
672 security Resources such as ACLs, credentials and Device owner transfer state.

673 **3.1.27**  
674 **Security Virtual Resource (SVR)**  
675 a resource supporting security features.  
676 Note 1 to entry: For a list of all the SVRs please see Clause 13.

677 **3.1.28**  
678 **Server**  
679 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

680 **3.1.29**  
681 **Trust Anchor**  
682 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.  
683 a Device and an onboarding tool) can assume trust

684 **3.1.30**  
685 **Unique Authenticable Identifier**  
686 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is  
687 used to create the Device ID.  
688 Note 1 to entry: The ownership of a UAID may be authenticated by peer Devices.

689 **3.1.31**  
690 **Device Configuration Resource (DCR)**  
691 a Resource that is any of the following:  
692 a) a Discovery Core Resource, or  
693 b) a Security Virtual Resource, or  
694 c) a WiFi Easy Setup Resource, or  
695 d) a CoAP Cloud Conf Resource.

696 **3.1.32**  
697 **Non-Configuration Resource (NCR)**  
698 a Resource that is not a Device Configuration Resource (3.1.31).

699 Note 1 to entry: This includes – for example – all the OCF Resources defined in ISO/IEC 30118-4:2018, as well as all  
700 vendor-defined Resources.

701 **3.1.33**  
702 **Bridged Device**  
703 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

704 **3.1.34**  
705 **Bridged Protocol**  
706 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

707 **3.1.35**  
708 **OCF Bridge Device**  
709 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

710 **3.1.36**  
711 **Virtual Bridged Device**  
712 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

713 **3.1.37**  
714 **Virtual OCF Device**  
715 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

716 **3.1.38**  
717 **OCF Security Domain**  
718 set of onboarded OCF Devices that are provisioned with credentialing information for confidential  
719 communication with one another

720 **3.2 Abbreviated terms**

721 **3.2.1**  
722 **AC**  
723 Access Control

724 **3.2.2**  
725 **ACE**  
726 Access Control Entry

727 **3.2.3**  
728 **ACL**  
729 Access Control List

730 **3.2.4**  
731 **AES**  
732 Advanced Encryption Standard  
733 Note 1 to entry: See NIST FIPS 197, "Advanced Encryption Standard (AES)"

734 **3.2.5**  
735 **AMS**  
736 Access Management Service

737 **3.2.6**  
738 **CMS**  
739 Credential Management Service

740 **3.2.7**  
741 **CRUDN**  
742 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

743 **3.2.8**  
744 **CSR**  
745 Certificate Signing Request

746 **3.2.9**  
747 **CVC**  
748 Code Verification Certificate

749 **3.2.10**  
750 **ECC**  
751 Elliptic Curve Cryptography

752 **3.2.11**  
753 **ECDSA**  
754 Elliptic Curve Digital Signature Algorithm

755 **3.2.12**  
756 **EKU**  
757 Extended Key Usage

758 **3.2.13**  
759 **EPC**  
760 Embedded Platform Credential

761 **3.2.14**  
762 **EPK**  
763 Embedded Public Key

764 **3.2.15**  
765 **DOTS**  
766 Device Ownership Transfer Service

767 **3.2.16**  
768 **DPKP**  
769 Dynamic Public Key Pair

770 **3.2.17**  
771 **ID**  
772 Identity/Identifier

773 **3.2.18**  
774 **JSON**  
775 See ISO/IEC 30118-1:2018.

776 **3.2.19**  
777 **JWS**  
778 JSON Web Signature.

779 Note 1 to entry: See IETF RFC 7515, "JSON Web Signature (JWS)"

780 **3.2.20**  
781 **KDF**  
782 Key Derivation Function

783 **3.2.21**  
784 **MAC**  
785 Message Authentication Code

786 **3.2.22**  
787 **MITM**  
788 Man-in-the-Middle

789 **3.2.23**  
790 **NVRAM**  
791 Non-Volatile Random-Access Memory

792 **3.2.24**  
793 **OC**  
794 Owner Credential

795 **3.2.25**  
796 **OCSP**  
797 Online Certificate Status Protocol

798 **3.2.26**  
799 **OBT**  
800 Onboarding Tool

801 **3.2.27**  
802 **OID**  
803 Object Identifier

804 **3.2.28**  
805 **OTM**  
806 Owner Transfer Method

807 **3.2.29**  
808 **OOB**  
809 Out of Band

810 **3.2.30**  
811 **OWASP**  
812 Open Web Application Security Project. See <https://www.owasp.org/>

813 **3.2.31**  
814 **PE**  
815 Policy Engine

816 **3.2.32**  
817 **PIN**  
818 Personal Identification Number

819 **3.2.33**  
820 **PPSK**  
821 PIN-authenticated pre-shared key

822 **3.2.34**  
823 **PRF**  
824 Pseudo Random Function

825 **3.2.35**  
826 **PSI**  
827 Persistent Storage Interface

828 **3.2.36**  
829 **PSK**  
830 Pre Shared Key

831 **3.2.37**  
832 **RBAC**  
833 Role Based Access Control

834 **3.2.38**  
835 **RM**  
836 Resource Manager

837 **3.2.39**  
838 **RNG**  
839 Random Number Generator

840 **3.2.40**  
841 **SACL**  
842 Signed Access Control List

843 **3.2.41**  
844 **SBAC**  
845 Subject Based Access Control

846 **3.2.42**  
847 **SEE**  
848 Secure Execution Environment

849 **3.2.43**  
850 **SRM**  
851 Secure Resource Manager

852 **3.2.44**  
853 **SVR**  
854 Security Virtual Resource

855 **3.2.45**  
856 **SW**  
857 Software

858 **3.2.46**  
859 **UAID**  
860 Unique Authenticable Identifier

861 **3.2.47**  
862 **URI**  
863 See ISO/IEC 30118-1:2018.

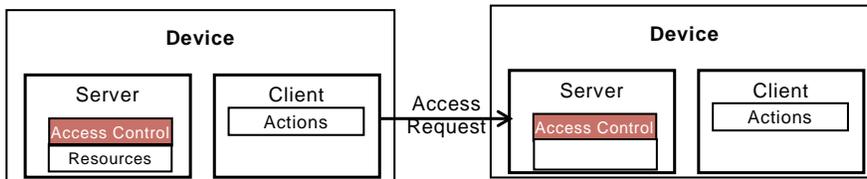
## 864 **4 Document Conventions and Organization**

### 865 **4.1 Conventions**

866 This document defines Resources, protocols and conventions used to implement security for OCF  
867 core framework and applications.

868 For the purposes of this document, the terms and definitions given in ISO/IEC 30118-1:2018 apply.

869 Figure 1 depicts interaction between OCF Devices.



870

871

**Figure 1 – OCF Interaction**

872 Devices may implement a Client role that performs Actions on Servers. Actions access Resources managed by Servers. The OCF stack enforces access policies on Resources. End-to-end Device interaction can be protected using session protection protocol (e.g. DTLS) or with data encryption methods.

#### 876 4.2 Notation

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

879 **Required** (or **shall** or **mandatory**).

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

883 **Recommended** (or **should**).

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

890 **Allowed** (may or allowed).

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

893 **Conditionally allowed** (CA)

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

896 **Conditionally required** (CR)

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

900 **DEPRECATED**

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

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

908 Words that are emphasized are printed in italic.

909 **4.3 Data types**

910 See ISO/IEC 30118-1:2018.

911 **4.4 Document structure**

912 Informative clauses may be found in the Overview clauses, while normative clauses fall outside of  
913 those clauses.

914 The Security Specification may use the oneM2M Release 3 Specifications,  
915 <http://www.onem2m.org/technical/published-drafts>

916 OpenAPI specification as the API definition language. The mapping of the CRUDN actions is  
917 specified in ISO/IEC 30118-1:2018.

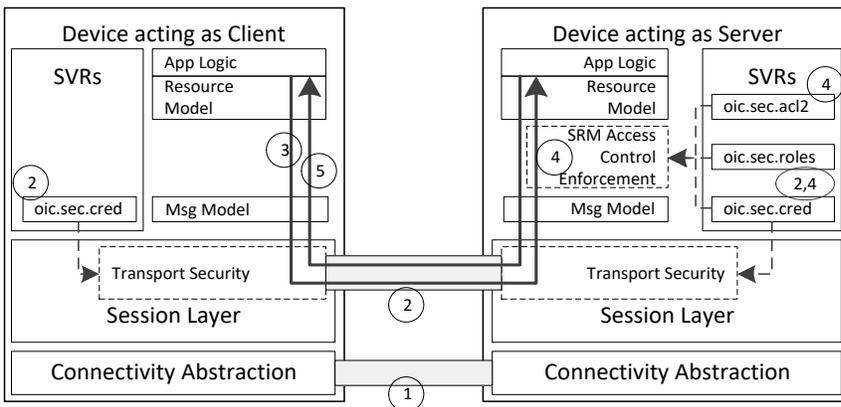
918

919 **5 Security Overview**

920 **5.1 Preamble**

921 This is an informative clause. The goal for the OCF security architecture is to protect the Resources  
922 and all aspects of HW and SW that are used to support the protection of Resource. From OCF  
923 perspective, a Device is a logical entity that conforms to the OCF documents. In an interaction  
924 between the Devices, the Device acting as the Server holds and controls the Resources and  
925 provides the Device acting as a Client with access to those Resources, subject to a set of security  
926 mechanisms. The Platform, hosting the Device may provide security hardening that will be required  
927 for ensuring robustness of the variety of operations described in this document.

928 The security theory of operation is depicted in Figure 2 and described in the following steps.



929  
930

931

**Figure 2 – OCF Layers**

- 932 1) The Client establishes a network connection to the Server (Device holding the Resources). The  
933 connectivity abstraction layer ensures the Devices are able to connect despite differences in  
934 connectivity options.
- 935 2) The Devices (e.g. Server and Client) exchange messages either with or without a mutually-  
936 authenticated secure channel between the two Devices.
- 937 a) The oic.sec.cred Resource on each Devices holds the credentials used for mutual  
938 authentication and (when applicable) certificate validation.
  - 939 b) Messages received over a secured channel are associated with a deviceUUID. In the case  
940 of a certificate credential, the deviceUUID is in the certificate received from the other Device.  
941 In the case of a symmetric key credential, the deviceUUID is configured with the credential  
942 in the oic.sec.cred Resource.
  - 943 c) The Server can associate the Client with any number of roleid. In the case of mutual  
944 authentication using a certificate, the roleid (if any) are provided in role certificates; these

945 are configured by the Client to the Server. In the case of a symmetric key, the allowed roleid  
946 (if any) are configured with the credential in the oic.sec.cred.

947 d) Requests received by a Server over an unsecured channel are treated as anonymous and  
948 not associated with any deviceUUID or roleid.

949 3) The Client submits a request to the Server.

950 4) The Server receives the request.

951 a) If the request is received over an unsecured channel, the Server treats the request as  
952 anonymous and no deviceUUID or roleid are associated with the request.

953 b) If the request is received over a secure channel, then the Server associates the deviceUUID  
954 with the request, and the Server associates all valid roleid of the Client with the request.

955 c) The Server then consults the Access Control List (ACL), and looks for an ACL entry  
956 matching the following criteria:

957 i) The requested Resource matches a Resource reference in the ACE

958 ii) The requested operation is permitted by the "permissions" of the ACE, and

959 iii) The "subjectUUID" contains either one of a special set of wildcard values or, if the  
960 Device is not anonymous, the subject matches the Client Deviceid associated with the  
961 request or a valid roleid associated with the request. The wildcard values match either  
962 all Devices communicating over an authenticated and encrypted session, or all Devices  
963 communicating over an unauthenticated and unencrypted session.

964 If there is a matching ACE, then access to the Resource is permitted; otherwise access  
965 is denied. Access is enforced by the Server's Secure Resource manager (SRM).

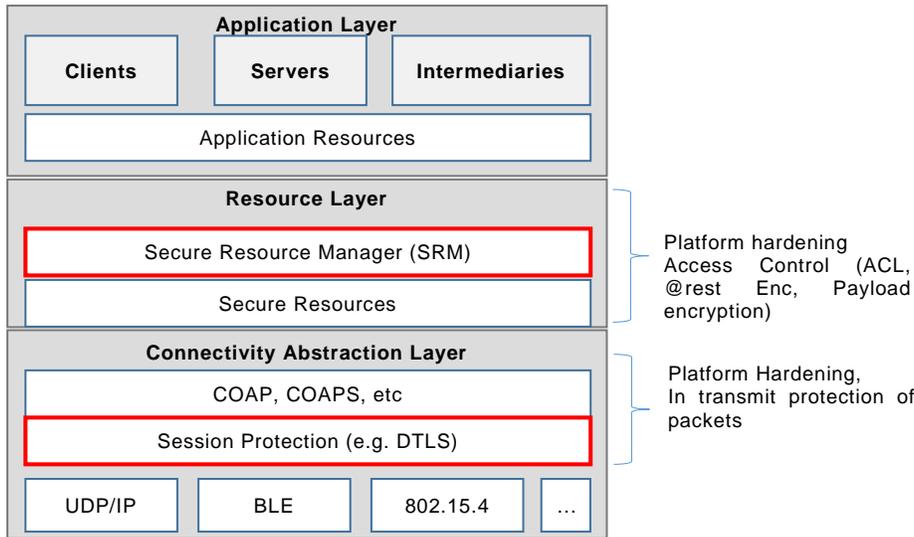
966 5) The Server sends a response back to the Client.

967 Resource protection includes protection of data both while at rest and during transit. Aside from  
968 access control mechanisms, the OCF Security Specification does not include specification of  
969 secure storage of Resources, while stored at Servers. However, at rest protection for security  
970 Resources is expected to be provided through a combination of secure storage and access control.  
971 Secure storage can be accomplished through use of hardware security or encryption of data at rest.  
972 The exact implementation of secure storage is subject to a set of hardening requirements that are  
973 specified in Clause 14 and may be subject to certification guidelines.

974 Data in transit protection, on the other hand, will be specified fully as a normative part of this  
975 document. In transit protection may be afforded at the resource layer or transport layer. This  
976 document only supports in transit protection at transport layer through use of mechanisms such as  
977 DTLS.

978 NOTE DTLS will provide packet by packet protection, rather than protection for the payload as whole. For instance, if  
979 the integrity of the entire payload as a whole is required, separate signature mechanisms must have already been in  
980 place before passing the packet down to the transport layer.

981 Figure 3 depicts OCF Security Enforcement Points.



982  
983  
984 **Figure 3 – OCF Security Enforcement Points**

985 A Device is authorized to communicate with an OCF Cloud if a trusted Mediator has provisioned  
986 the Device.

- 987 – Device and Mediator connect over DTLS using "/oic/sec/cred"
- 988 – Device is provisioned by Mediator with following information:
  - 989 – the URI of OCF Cloud
  - 990 – Token that can be validated by the OCF Cloud
  - 991 – UUID of the OCF Cloud

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

## 995 5.2 Access Control

996 The OCF framework assumes that Resources are hosted by a Server and are made available to  
997 Clients subject to access control and authorization mechanisms. The Resources at the end point  
998 are protected through implementation of access control, authentication and confidentiality  
999 protection. This clause provide an overview of Access Control (AC) through the use of ACLs.  
1000 However, AC in the OCF stack is expected to be transport and connectivity abstraction layer  
1001 agnostic.

1002 Implementation of access control relies on a-priori definition of a set of access policies for the  
1003 Resource. The policies may be stored by a local ACL or an Access Management Service (AMS) in  
1004 form of Access Control Entries (ACE). Two types of access control mechanisms can be applied:

- 1005 – Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of  
1006 requestor) of the requesting entity against the subject included in the policy defined for  
1007 Resource. Asserting the identity of the requestor requires an authentication process.

1008 – Role-based Access Control (RBAC), where each ACE will match a role identifier included in the  
1009 policy for the Resource to a role identifier associated with the requestor

1010 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only  
1011 local references and there is an ACE matching the Atomic Measurement Resource which permits  
1012 the request, then the corresponding requests to linked Resources are permitted by the OCF Server.  
1013 The present paragraph shall apply to any Resource Type based on the Atomic Measurement  
1014 Resource Type.

1015 NOTE The definition of an Atomic Measurement Resource prohibits direct access to the linked Resources. The nature  
1016 of an Atomic Measurement also prohibits updating the "links" to add or remove Resources. Consequently, there is no risk  
1017 of privilege escalation when using the ACE of an Atomic Measurement Resource to govern access to its linked  
1018 Resources.

1019 If an OCF Server receives a batch request to a Collection Resource containing only local references  
1020 and there is an ACE matching the Collection Resource which permits the request, then the  
1021 corresponding requests to linked Resources are permitted by the OCF Server. The present  
1022 paragraph shall apply to any Resource Type based on the Collection Resource Type.

1023 NOTE This implies that the ACEs of the Collection Resource permit access to all the Collection's linked Resources via  
1024 the batch OCF Interface, even if there are no ACEs permitting direct access to some or all the linked Resources. If not  
1025 tightly governed, this could lead to privilege escalation. Restrictions on the use of Collection Resources have been  
1026 provided in ISO/IEC 30118-1:2018 to mitigate the risk of privilege escalation. For example, ISO/IEC 30118-1:2018  
1027 prohibits updating "links" of a Collection Resource with the intent of obtaining access to the added Resource according  
1028 to the ACEs of the Collection, when access to the Resource would have otherwise been denied.

1029 In the OCF access control model, access to a Resource instance requires an associated access  
1030 control policy. This means, each Device acting as Server, needs to have an ACE permitting access  
1031 to each Resource it is protecting. This criterion can be satisfied for a Resource A if there is an ACE  
1032 permitting batch requests to access Resource B containing a Link to Resource A, even if there are  
1033 no ACEs permitting requests which access Resource A directly. Examples of the Resource Type  
1034 for Resource B is the Atomic Measurement Resource Type and the Collection Resource Type. The  
1035 lack of an ACE permitting access to a Resource, either directly or via a Link results in the Resource  
1036 being inaccessible.

1037 The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested  
1038 Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier or  
1039 (3) wildcard. The way in which the client connects to the server may be relevant context for making  
1040 access control decisions. Wildcard matching on authenticated vs. unauthenticated and encrypted  
1041 vs. unencrypted connection allows an access policy to be broadly applied to subject classes.

1042 Example Wildcard Matching Policy:

```
1043 "aclist2": [  
1044 {  
1045   "subject": {"conntype": "anon-clear" },  
1046   "resources": [  
1047     { "wc": "*" }  
1048   ],  
1049   "permission": 31  
1050 },  
1051 {  
1052   "subject": {"conntype": "auth-crypt" },  
1053   "resources": [  
1054     { "wc": "*" }  
1055   ],  
1056   "permission": 31
```

1057     ),  
1058     ]  
1059     Details of the format for ACL are defined in Clause 12. The ACL is composed of one or more ACEs.  
1060     The ACL defines the access control policy for the Devices.

1061     ACL Resource requires the same security protection as other sensitive Resources, when it comes  
1062     to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW and  
1063     SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may have  
1064     different scoping levels and thus hardening needs to be specially considered for each scoping level.  
1065     For instance a physical device may host multiple Device implementations and thus secure storage,  
1066     usage and isolation of ACLs for different Servers on the same Device needs to be considered.

## 1067     **5.2.1     ACL Architecture**

### 1068     **5.2.1.1     ACL Architecture General**

1069     The Server examines the Resource(s) requested by the client before processing the request. The  
1070     access control resources ~~(e.g. "/oic/sec/acl", "/oic/sec/acl2")~~ are is searched to find one or more  
1071     ACE entries that match the requestor and the requested Resources. If a match is found then  
1072     permission and period constraints are applied. If more than one match is found then the logical  
1073     UNION of permissions is applied to the overlapping periods.

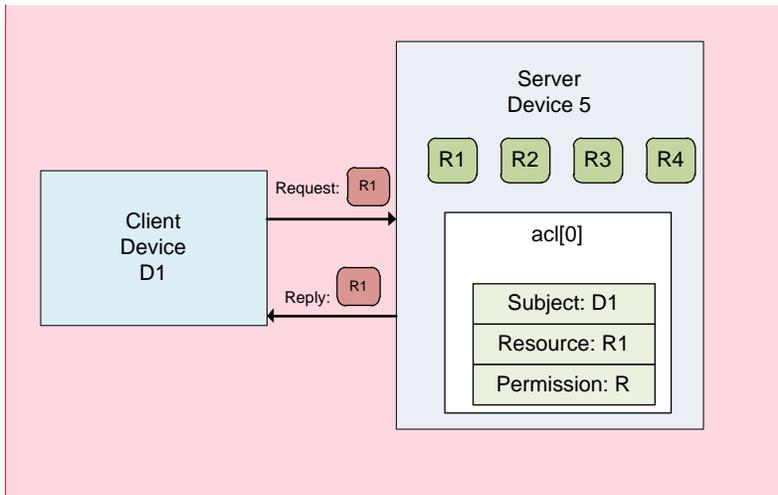
1074     The server uses the connection context to determine whether the subject has authenticated or not  
1075     and whether data confidentiality has been applied or not. Subject matching wildcard policies can  
1076     match on each aspect. If the user has authenticated, then subject matching may happen at  
1077     increased granularity based on role or device identity.

1078     Each ACE contains the permission set that will be applied for a given Resource requestor.  
1079     Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY  
1080     (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more  
1081     roles. Devices may acquire elevated access permissions when asserting a role. For example, an  
1082     ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally  
1083     accessible.

### 1084     **5.2.1.2     Use of local ACLs**

1085     Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control  
1086     processing than remote ACL processing by an AMS.  
1087     The following use cases describe the operation of access control

1088     Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4). Client  
1089     Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0] corresponds to  
1090     Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives access to  
1091     Resource R1 because the local ACL /oic/sec/acl~~2~~/0 matches the request.



Commented [OA1]: Change acl to acl2

Figure 4 – Use case-1 showing simple ACL enforcement

1092

1093

1094 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL  
 1095 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.

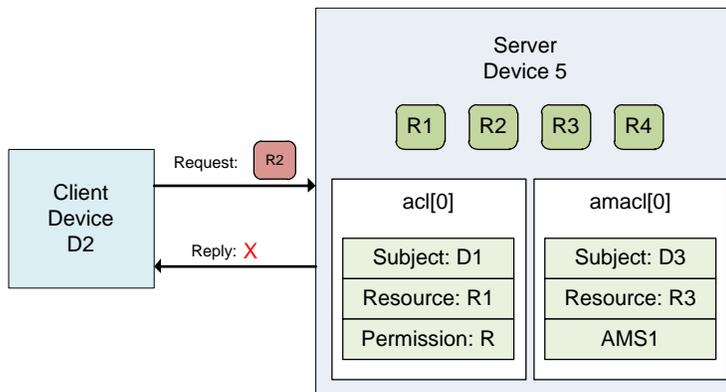


Figure 5 – Use case 2: A policy for the requested Resource is missing

1096

1097

1098 **5.2.1.3 Use of AMS**

1099 AMS improves ACL policy management. However, they can become a central point of failure. Due  
 1100 to network latency overhead, ACL processing may be slower through an AMS.

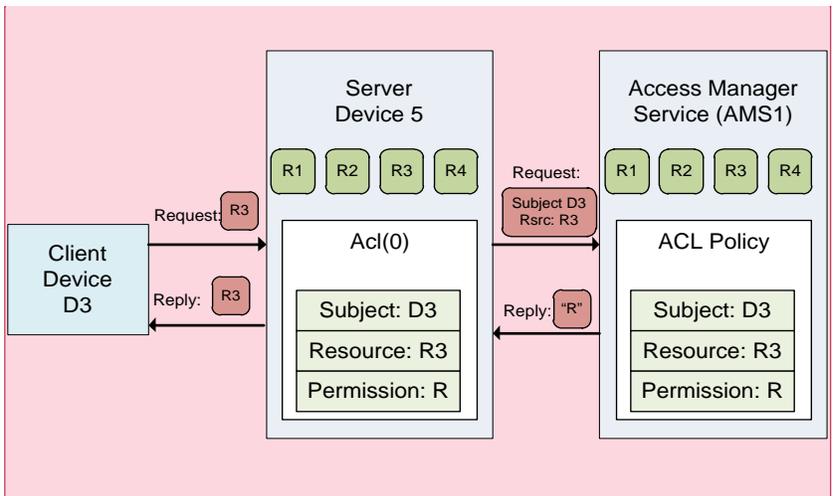
1101 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The  
 1102 Server shall determine which ACL mechanism to use for which Resource set. The /oic/sec/amacl  
 1103 Resource is an ACL structure that specifies which Resources will use an AMS to resolve access  
 1104 decisions. The /oic/sec/amacl may be used in concert with local ACLs ("/oic/sec/acl2").

1105 The AMS is authenticated by referencing a credential issued to the device identifier contained in  
 1106 "/oic/sec/acl2.rowneruid".

1107 The Server Device may proactively open a connection to the AMS using the Device ID found in  
 1108 /oic/sec/acl2.rowneruid. Alternatively, the Server may reject the Resource access request with an  
 1109 error, ACCESS\_DENIED\_REQUIRES\_SACL that instructs the requestor to obtain a suitable ACE  
 1110 policy using a SACL Resource /oic/sec/sacl. The /oic/sec/sacl signature may be validated using  
 1111 the credential Resource associated with the /oic/sec/acl2.rowneruid.

1112 The following use cases describe access control using the AMS:

1113 Use Case 3: As depicted in Figure 6, Device D3 requests and receives access to Resource R3 with  
 1114 permission Perm1 because the /oic/sec/amacl/0 matches a policy to consult the Access Manager  
 1115 Server AMS1 service



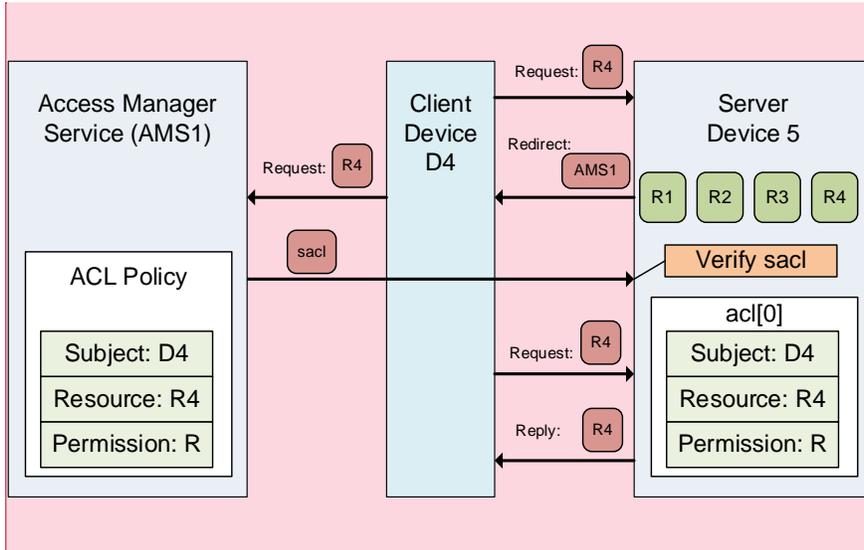
Commented [OA2]: Acl2

1116  
 1117 **Figure 6 – Use case-3 showing AMS supported ACL**

1118 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from Server  
 1119 Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1 by  
 1120 returning an error identifying AMS1 as a /oic/sec/sacl Resource issuer. Device D4 obtains Sacl1  
 1121 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the /oic/sec/sacl  
 1122 Resource and evaluates the ACE policy that grants Perm2 access.

1123 ACE redirection may occur when D4 receives an error result with reason code indicating no match  
 1124 exists (i.e. ACCESS\_DENIED\_NO\_ANCE). D4 reads the /oic/sec/acl2 Resource to find the  
 1125 rowneruid which identifies the AMS and then submits a request to be provisioned, in this example  
 1126 the AMS chooses to supply a SACL Resource, however it may choose to re-provision the local ACL  
 1127 Resources ~~/oic/sec/acl~~ and ~~/oic/sec/acl2~~. The request is reissued subsequently. D4 is presumed  
 1128 to have been introduced to the AMS as part of Device onboarding or through subsequent credential  
 1129 provisioning actions.

1130 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.



Commented [OA3]: Acl to acl2

1131  
1132 **Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS**

1133 **5.2.2 Access Control Scoping Levels**

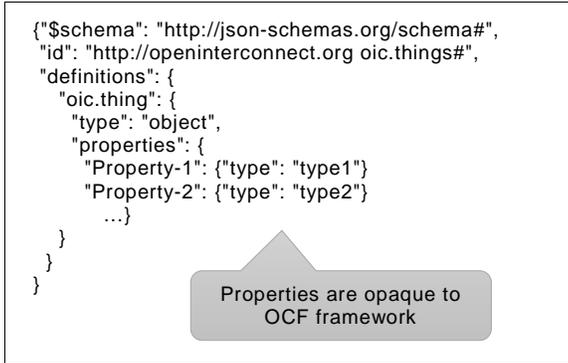
1134 **Group Level Access** - Group scope means applying AC to the group of Devices that are grouped  
1135 for a specific context. Group Level Access means all group members have access to group data  
1136 but non-group members must be granted explicit access. Group level access is implemented using  
1137 Role Credentials and/or connection type

1138 **OCF Device Level Access** – OCF Device scope means applying AC to an individual Device, which  
1139 may contain multiple Resources. Device level access implies accessibility extends to all Resources  
1140 available to the Device identified by Device ID. Credentials used for AC mechanisms at Device are  
1141 OCF Device-specific.

1142 **OCF Resource Level Access** – OCF Resource level scope means applying AC to individual  
1143 Resources. Resource access requires an ACL that specifies how the entity holding the Resource  
1144 (Server) shall make a decision on allowing a requesting entity (Client) to access the Resource.

1145 **Property Level Access** - Property level scope means applying AC only to an individual Property  
1146 Property level access control is only achieved by creating a Resource that contains a single  
1147 Property.

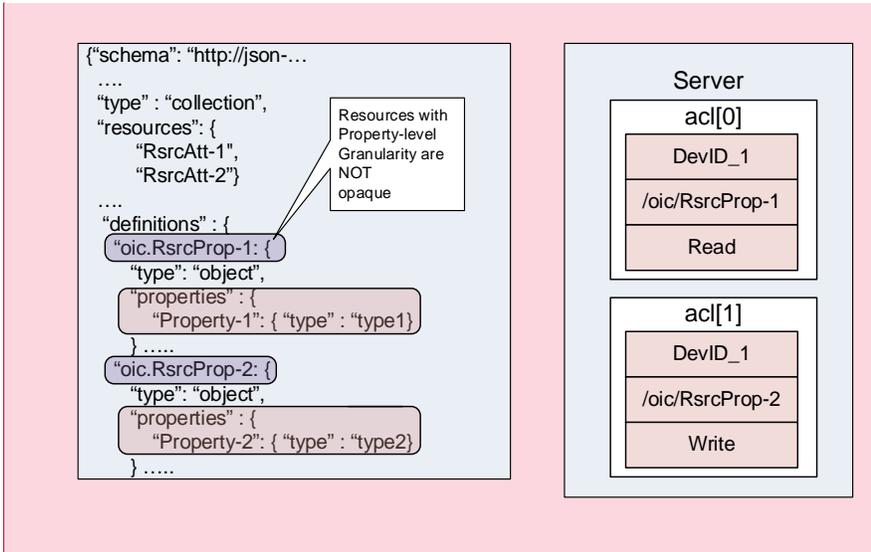
1148 Controlling access to static Resources where it is impractical to redesign the Resource, it may  
1149 appropriate to introduce a collection Resource that references the child Resources having separate  
1150 access permissions. An example is shown Figure 8, where an "oic.thing" Resource has two  
1151 properties: Property-1 and Property-2 that would require different permissions.



1152

1153 **Figure 8 – Example Resource definition with opaque Properties**

1154 Currently, OCF framework treats property level information as opaque; therefore, different  
 1155 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1  
 1156 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into  
 1157 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be  
 1158 achieved through use of Resource-level ACLs.



1159

1160 **Figure 9 – Property Level Access Control**

Commented [OA4]: Acl to acl2

1161 **5.3 Onboarding Overview**

1162 **5.3.1 Onboarding General**

1163 Before a Device becomes operational in an OCF environment and is able to interact with other  
1164 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to  
1165 configure the ownership where the legitimate user that owns/purchases the Device uses an  
1166 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to  
1167 establish ownership. Once ownership is established, the OBT becomes the mechanism through  
1168 which the Device can then be provisioned, at the end of which the Device becomes operational  
1169 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an  
1170 OCF Device.

1171 Figure 10 depicts Onboarding Overview.

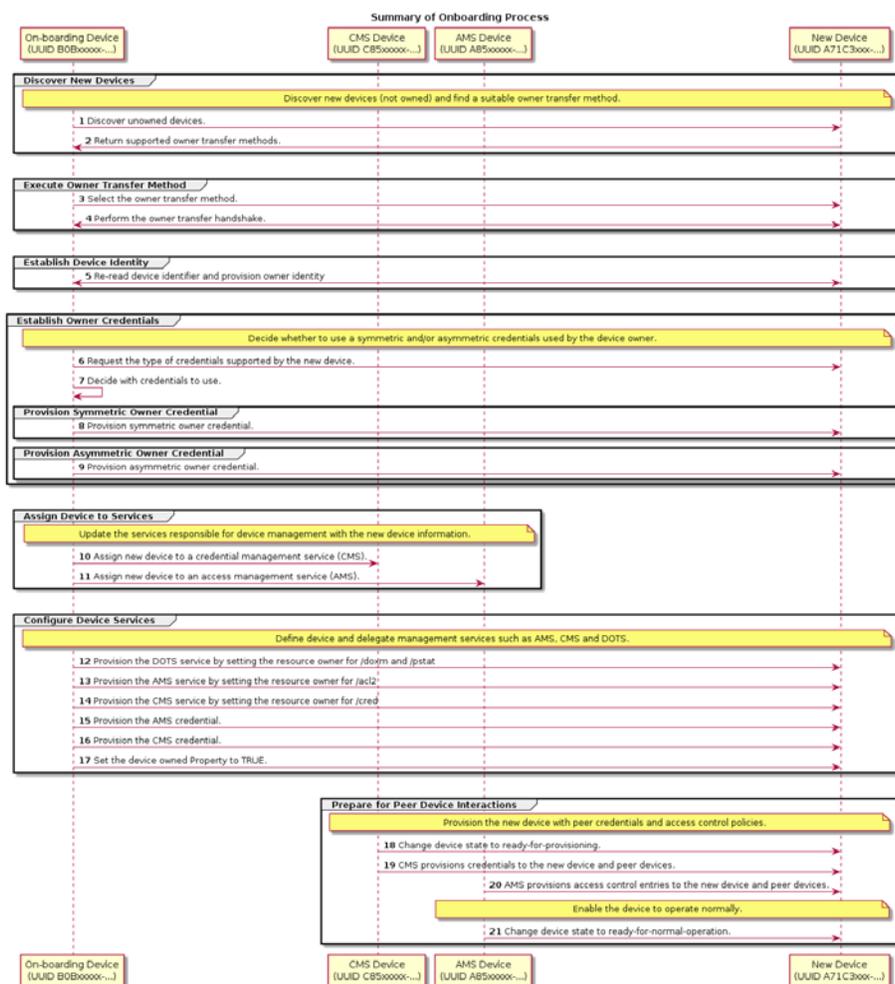


Figure 10 – Onboarding Overview

1172  
1173

1174 This clause explains the onboarding and security provisioning process but leaves the provisioning  
 1175 of non-security aspects to other OCF documents. In the context of security, all Devices are required  
 1176 to be provisioned with minimal security configuration that allows the Device to securely  
 1177 interact/communicate with other Devices in an OCF environment. This minimal security  
 1178 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified  
 1179 in 7.5.

1180 Onboarding and provisioning implementations could utilize services defined outside this document,  
 1181 it is expected that in using other services, trust between the device being onboarded and the  
 1182 various tools is not transitive. This implies that the device being onboarded will individually

1183 authenticate the credentials of each and every tool used during the onboarding process; that the  
1184 tools not share credentials or imply a trust relationship where one has not been established.

### 1185 **5.3.2 OnBoarding Steps**

1186 The flowchart in Figure 11 shows the typical steps that are involved during onboarding. Although  
1187 onboarding may include a variety of non-security related steps, the diagram focus is mainly on the  
1188 security related configuration to allow a new Device to function within an OCF environment.  
1189 Onboarding typically begins with the Device getting "owned" by the legitimate user/system followed  
1190 by configuring the Device for the environment that it will operate in. This would include setting  
1191 information such as who can access the Device and what actions can be performed as well as what  
1192 permissions the Device has for interacting with other Devices.

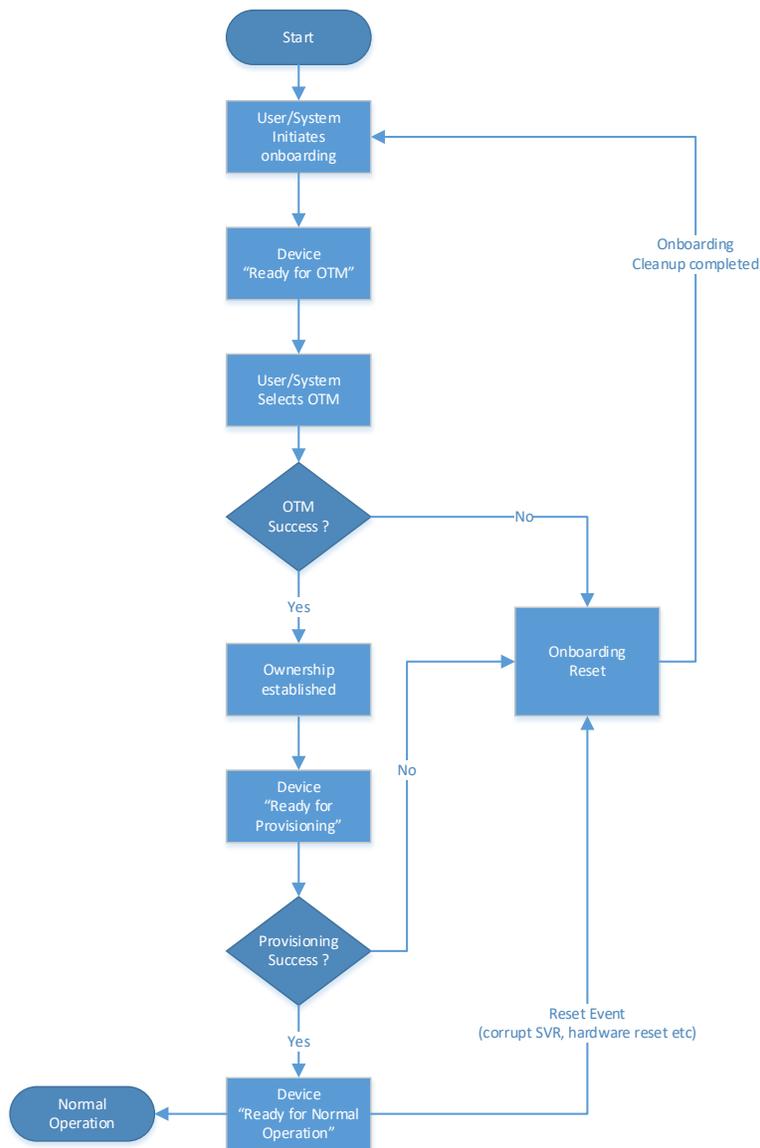


Figure 11 – OCF Onboarding Process

5.3.3 Establishing a Device Owner

The objective behind establishing Device ownership is to allow the legitimate user that owns/purchased the Device to assert itself as the owner and manager of the Device. This is done Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved

1198 through the use of an OBT that includes the creation of an ownership context between the new  
1199 Device and the OBT tool and asserts operational control and management of the Device. The OBT  
1200 can be considered a logical entity hosted by tools/ Servers such as a network management console,  
1201 a device management tool, a network-authoring tool, a network provisioning tool, a home gateway  
1202 device, or a home automation controller. A physical device hosting the OBT will be subject to some  
1203 security hardening requirements, thus preserving integrity and confidentiality of any credentials  
1204 being stored. The tool/Server that establishes Device ownership is referred to as the OBT.

1205 The OBT uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term  
1206 owner transfer is used since it is assumed that even for a new Device, the ownership is transferred  
1207 from the manufacturer/provider of the Device to the buyer/legitimate user of the new Device.

1208 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.  
1209 Owner transfer establishes the following

1210 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"  
1211 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during  
1212 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruid property  
1213 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the  
1214 identity of Device as part of ownership transfer.

1215 – The Device owner establishes trust in the Device through the OTM.

1216 – Preparing the Device for provisioning by providing credentials that may be needed.

#### 1217 **5.3.4 Provisioning for Normal Operation**

1218 Once the Device has the necessary information to initiate provisioning, the next step is to provision  
1219 additional security configuration that allows the Device to become operational. This can include  
1220 setting various parameters and may also involve multiple steps. Also provisioning of ACL's for the  
1221 various Resources hosted by the Server on the Device is done at this time. The provisioning step  
1222 is not limited to this stage only. Device provisioning can happen at multiple stages in the Device's  
1223 operational lifecycle. However specific security related provisioning of Resource and Property state  
1224 would likely happen at this stage at the end of which, each Device reaches the Onboarded Device  
1225 "Ready for Normal Operation" State. The "Ready for Normal Operation" State is expected to be  
1226 consistent and well defined regardless of the specific OTM used or regardless of the variability in  
1227 what gets provisioned. However individual OTM mechanisms and provisioning steps may specify  
1228 additional configuration of Resources and Property states. The minimal mandatory configuration  
1229 required for a Device to be in "Ready for Normal Operation" state is specified in 8.

#### 1230 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview**

1231 As mentioned in the start of Clause 5, communication between a Device and OCF Cloud is subject  
1232 to different criteria in comparison to Devices which are within a single local network. The Device is  
1233 configured in order to connect to the OCF Cloud by a Mediator as specified in the CoAPCloudConf  
1234 Resource clauses in ISO/IEC 30118-8:2018. Provisioning includes the remote connectivity and  
1235 local details such as URL where the OCF Cloud hosting environment can be found and the OCF  
1236 Cloud verifiable Access Token.

#### 1237 **5.3.6 OCF Compliance Management System**

1238 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that  
1239 provides Certification status and information for OCF Devices.

1240 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:  
1241 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1242 The OBT shall possess the Root Certificate needed to enable https connection to the URI  
1243 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1244 The OBT should periodically refresh its copy of the CPL via the URI  
1245 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security  
1246 Domain owner policy requirements.

## 1247 **5.4 Provisioning**

### 1248 **5.4.1 Provisioning General**

1249 In general, provisioning may include processes during manufacturing and distribution of the Device  
1250 as well as processes after the Device has been brought into its intended environment (parts of  
1251 onboarding process). In this document, security provisioning includes, processes after ownership  
1252 transfer (even though some activities during ownership transfer and onboarding may lead to  
1253 provisioning of some data in the Device) configuration of credentials for interacting with  
1254 provisioning services, configuration of any security related Resources and credentials for dealing  
1255 with any services that the Device need to contact later on.

1256 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to  
1257 be provisioned with proper security credentials and parameters for regular operation. These  
1258 parameters can include:

- 1259 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1260 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the  
1261 same OBT, but may be part of AMS in future.

1262 As mentioned, to accommodate a scalable and modular design, these functions are considered as  
1263 services that in future could be deployed as separate servers. Currently, the deployment assumes  
1264 that these services are all deployed as part of a OBT. Regardless of physical deployment scenario,  
1265 the same security-hardening requirement) applies to any physical server that hosts the tools and  
1266 security provisioning services discussed here.

1267 Devices are *aware* of their security provisioning status. Self-awareness allows them to be proactive  
1268 about provisioning or re-provisioning security Resources as needed to achieve the devices  
1269 operational goals.

### 1270 **5.4.2 Provisioning other services**

1271 To be able to support the use of potentially different device management service hosts, each Device  
1272 Secure Virtual Resource (SVR) has an associated Resource owner identified in the Resource's  
1273 `rowneruid` Property.

1274 The DOTS shall update the `rowneruid` Property of the `/oic/sec/doxm` and `/oic/sec/pstat` resources  
1275 with the DOTS resource owner identifier.

1276 The DOTS shall update the `rowneruid` Property of the `/oic/sec/cred` resource with the CMS  
1277 resource owner identifier.

1278 The DOTS shall update the `rowneruid` Property of the `/oic/sec/acl2` resource with the AMS  
1279 resource owner identifier

1280 When these OCF Services are configured, the Device may proactively request provisioning and  
1281 verify provisioning requests are authorized. The DOTS shall provision credentials that enable  
1282 secure connections between OCF Services and the new Device. The DOTS may initiate client-  
1283 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed  
1284 provisioning by setting `tm` Property of the `/oic/sec/pstat` Resource.

### 1285 **5.4.3 Provisioning Credentials for Normal Operation**

1286 The `/oic/sec/cred` Resource supports multiple types of credentials including:

- 1287 – Pairwise symmetric keys

1288 – Group symmetric keys

1289 – Certificates

1290 – Raw asymmetric keys

1291 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS  
1292 credential provisioned by the DOTS.

1293 The following example describes how a Device updates a symmetric key credential involving a peer  
1294 Device. The Device discovers the credential to be updated; for example, a secure connection  
1295 attempt fails. The Device requests its CMS to supply the updated credential. The CMS returns an  
1296 updated symmetric key credential. The CMS updates the corresponding symmetric key credential  
1297 on the peer Device.

#### 1298 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1299 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)  
1300 asserted by the Client requesting the Resource and compare that role identifier(s) with the  
1301 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned with  
1302 one or more role credentials.

1303 Each Device holds the role information as a Property within the credential Resource.

1304 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a  
1305 certificate role credential.

1306 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned  
1307 for a client, access to a Resource is granted if it would be granted under any of the roles.

#### 1308 **5.4.5 ACL provisioning**

1309 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.  
1310 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL  
1311 policy by updating the Device's ACL Resources.

1312 The AMS shall digitally sign an ACL as part of issuing a /oic/sec/sacl Resource if the Device  
1313 supports the /oic/sec/sacl Resource. The public key used by the Device to verify the signature shall  
1314 be provisioned by the CMS as needed. A /oic/sec/cred Resource with an asymmetric key type or  
1315 signed asymmetric key type is used. The PublicData Property contains the AMS's public key.

#### 1316 **5.5 Secure Resource Manager (SRM)**

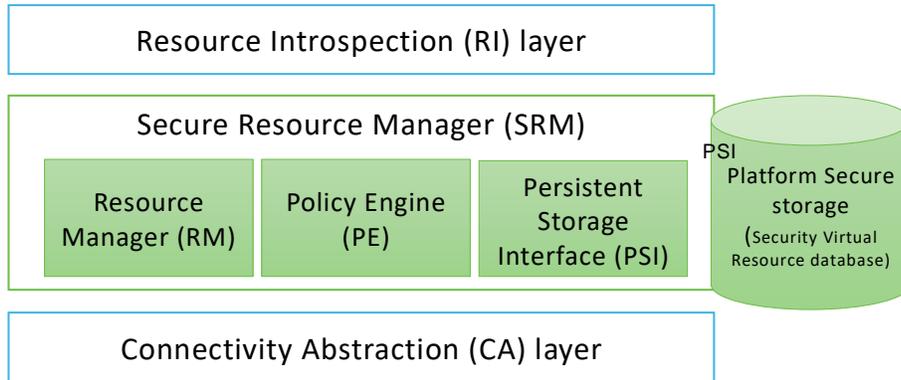
1317 SRM plays a key role in the overall security operation. In short, SRM performs both management  
1318 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3  
1319 main functional elements:

1320 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using PSI)  
1321 as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3) Responding  
1322 to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a format that is  
1323 consistent with device-specific data store format. However, the RM will use JSON format to  
1324 marshal SVR data structures before be passed to PSI for storage, or travel off-device.

1325 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control  
1326 policies responds to the requests with either "ACCESS\_GRANTED" or "ACCESS\_DENIED". To  
1327 make the access decisions, the PE consults the appropriate ACL and looks for best Access  
1328 Control Entry (ACE) that can serve the request given the subject (Device or role) that was  
1329 authenticated by DTLS.

1330 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in  
1331 its own memory and storage. The SRM design is modular such that it may be implemented in  
1332 the Platform's secure execution environment; if available.

1333 Figure 12 depicts OCF's SRM Architecture.



1334

1335 **Figure 12 – OCF's SRM Architecture**

1336 **5.6 Credential Overview**

1337 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional  
1338 communication. Credentials can be symmetric or asymmetric. Each device stores secret and public  
1339 parts of its own credentials where applicable, as well as credentials for other devices that have  
1340 been provided by the DOTS or a CMS. These credentials are then used in the establishment of  
1341 secure communication sessions (e.g. using DTLS) to validate the identities of the participating  
1342 parties. Role credentials are used once an authenticated session is established, to assert one or  
1343 more roles for a device.

1344 Access Tokens are provided to an OCF Cloud once an authenticated session with an OCF Cloud  
1345 is established, to verify the User ID with which the Device is to be associated.

1346

1347 **6 Security for the Discovery Process**

1348 **6.1 Preamble**

1349 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,  
1350 called links) for the Resources hosted by the Server, complemented by attributes about those  
1351 Resources and possible further link relations. (in accordance to Clause 10 in ISO/IEC 30118-  
1352 1:2018)

1353 **6.2 Security Considerations for Discovery**

1354 When defining discovery process, care must be taken that only a minimum set of Resources are  
1355 exposed to the discovering entity without violating security of sensitive information or privacy  
1356 requirements of the application at hand. This includes both data included in the Resources, as well  
1357 as the corresponding metadata.

1358 To achieve extensibility and scalability, this document does not provide a mandate on  
1359 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on  
1360 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any  
1361 of the Resources.

1362 The /oic/sec/acl2 Resource contains ACL entries governing access to the Server hosted Resources.  
1363 (See 13.5)

1364 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery  
1365 process itself needs to be secured. This document sets the following requirements for the discovery  
1366 process:

- 1367 1) Providing integrity protection for discovered Resources.  
1368 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1369 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)  
1370 on the known "/oic/res" Resource.

1371 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a  
1372 Server cannot determine the identity of the requester. In such cases, a Server that wants to  
1373 authenticate the Client before responding can list the secure discovery URI (e.g.  
1374 coaps://IP:PORT/oic/res ) in the unsecured /oic/res Resource response. This means the secure  
1375 discovery URI is by default discoverable by any Client. The Client will then be required to send a  
1376 separate unicast request using DTLS to the secure discovery URI.

1377 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to  
1378 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN  
1379 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1380 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource  
1381 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1382 {  
1383   "aclist2": [  
1384     {  
1385       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1386       "resources": [{"href": "/door"}],  
1387       "permission": 2, // RETRIEVE  
1388       "aceid": 1  
1389     }  
1390   ],
```

```

1391     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1392   }
1393   {
1394     "aclist2": [
1395       {
1396         "subject": {"authority": "owner", "role": "owner"},
1397         "resources": [{"href": "/door"}],
1398         "permission": 2, // RETRIEVE
1399         "aceid": 2
1400       }
1401     ],
1402     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1403   }
1404   {
1405     "aclist2": [
1406       {
1407         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1408         "resources": [{"href": "/door/lock"}],
1409         "permission": 4, // UPDATE
1410         "aceid": 3
1411       }
1412     ],
1413     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1414   }
1415   {
1416     "aclist2": [
1417       {
1418         "subject": {"conntype": "anon-clear"},
1419         "resources": [{"href": "/light"}],
1420         "permission": 2, // RETRIEVE
1421         "aceid": 4
1422       }
1423     ],
1424     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1425   }

```

1426 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when  
1427 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will include  
1428 the URI of the "/door" Resource metadata. Client "d2" will have access to both the Resources.  
1429 ACE2 will prevent "d4" from update.

1430 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1431 [
1432   {
1433     "href": "/door",
1434     "rt": ["oic.r.door"],
1435     "if": ["oic.if.b", "oic.if"],

```

```
1436     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1437   }
1438 ]
1439 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:
1440 [
1441 {
1442   "href": "/door",
1443   "rt": ["oic.r.door"],
1444   "if": ["oic.if.b", "oic.ll"],
1445   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1446 },
1447 {
1448   "href": "/door/lock",
1449   "rt": ["oic.r.lock"],
1450   "if": ["oic.if.b"],
1451   "type": ["application/json", "application/exi+xml"]
1452 }
1453 ]
1454 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:
1455 [
1456 {
1457   "href": "/door/lock",
1458   "rt": ["oic.r.lock"],
1459   "if": ["oic.if.b"],
1460   "type": ["application/json", "application/exi+xml"],
1461   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1462 }
1463 ]
1464 Discovery results delivered to any device regarding d3's /oic/res Resource from the unsecure
1465 interface:
1466 [
1467 {
1468   "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1469   "href": "/light",
1470   "rt": ["oic.r.light"],
1471   "if": ["oic.if.s"]
1472 }
1473 ]
1474
```

1475 **7 Security Provisioning**

1476 **7.1 Device Identity**

1477 **7.1.1 General Device Identity**

1478 Each Device, which is a logical device, is identified with a Device ID.

1479 Devices shall be identified by a Device ID value that is established as part of device onboarding.  
1480 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. urn:uuid). Device IDs shall be  
1481 unique within the scope of operation of the corresponding OCF Security Domain, and should be  
1482 universally unique. The DOTS shall ensure Device ID of the new Device is unique within the scope  
1483 of the owner's OCF Security Domain. The DOTS shall verify the chosen new device identifier does  
1484 not conflict with Device IDs previously introduced into the OCF Security Domain.

1485 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"  
1486 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying  
1487 authentication credentials of a peer device.

1488 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of credentials,  
1489 both its own and other Device credentials, in the /oic/sec/cred Resource. The device ID can be  
1490 used to distinguish between a device's own credential, and credentials for other devices.  
1491 Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the device.

1492 Device ID shall be:

- 1493 – Unique
- 1494 – Immutable
- 1495 – Verifiable

1496 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the  
1497 device as described later in this clause.

1498 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The  
1499 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in the  
1500 device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1501 An OCF Platform may have a secure execution environment, which shall be used to secure unique  
1502 identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to provide  
1503 each Device with the appropriate and separate security.

1504 **7.1.2 Device Identity for Devices with UAID**

1505 **7.1.2.1 Unique Authenticable Identifier**

1506 When a manufacturer certificate is used with certificates chaining to an OCF root CA (as specified  
1507 in 7.1.2), the manufacturer shall include a Platform ID inside the certificate subject CN field. In  
1508 such cases, the device ID may be created according to the Unique Authenticable Identifier (UAID)  
1509 scheme defined in this clause.

1510 For identifying and protecting Devices, the Platform Secure Execution Environment (SEE) may opt  
1511 to generate new Dynamic Public Key Pair (DPKP) for each Device it is hosting, or it may opt to  
1512 simply use the same public key credentials embedded by manufacturer; Embedded Platform  
1513 Credential (EPC). In either case, the Platform SEE will use its Random Number Generator (RNG)  
1514 to create a device identity called UAID for each Device. The UAID is generated using either EPC  
1515 only or the combination of DPKP and EPC if both are available. When both are available, the  
1516 Platform shall use both key pairs to generate the UAID as described in this clause.

1517 The Device ID is formed from the device's public keys and associated OCF Cipher Suite. The  
1518 Device ID is formed by:

1519 1) Determining the OCF Cipher Suite of the Dynamic Public Key. The Cipher Suite curve must  
1520 match the usage of the AlgorithmIdentifier used in SubjectPublicKeyInfo as intended for use  
1521 with Device security mechanisms. Use the encoding of the CipherSuite as the 'csid' value in  
1522 the following calculations. If the OCF Cipher Suite for Dynamic Public key is different from the  
1523 ciphersuite indicated in the Platform certificate (EPC), the OCF Cipher Suite shall be used  
1524 below.

1525 2) From EPC extract the value of embedded public key. The value should correspond to the value  
1526 of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate. In the following we refer  
1527 to this as EPK. If the public key is extracted from a certificate, validate that the  
1528 AlgorithmIdentifier matches the expected value for the CipherSuite within the certificate.

1529 3) From DPKP, extract the value of the public key. The value should correspond to the value of  
1530 subjectPublicKey defined in SubjectPublicKeyInfo. In the following we refer to this as DPK.

1531 4) Using the hash for the Cipher Suite calculate:  $h = \text{hash}(\text{'uaid'} \mid \text{csid} \mid \text{EPK} \mid \text{DPK} \mid \text{<other\_info>})$   
1532 Other\_info could be 1) device type as indicated in "/oic/d" (could be read-only and set by  
1533 manufacturer), 2) in case there are two sets of public key pairs (one embedded, and one  
1534 dynamically generated), both public keys would be included.

1535 5) Truncate to 160 bits by taking the leftmost 160 bits of h UAID =  $h[0:16]$  # leftmost 16 octets

1536 6) Convert the binary UAID to a ASCII string by  $\text{USID} = \text{base27encode}(\text{UAID})$

```
1537 def base_N_encode(octets, alphabet):  
1538     long_int = string_to_int( octets )  
1539     text_out = ''  
1540     while long_int > 0:  
1541         long_int, remainder = divmod(long_int, len(alphabet))  
1542         text_out = alphabet[remainder] + text_out  
1543     return text_out
```

```
1544  
1545 b27chars = 'ABCDEFGHJKMNPQRTWXYZ2346789'  
1546 def b27encode(octet_string):  
1547     """Encode a octet string using 27 characters. """  
1548     return base_N_encode(octet_string, _b27chars )
```

1549 7) Append the string value of USID to "urn:usid:" to form the final string value of the Device ID  
1550 urn:usid:ABXW....

1551 Whenever the public key is encoded the format described in IETF RFC 7250 for  
1552 SubjectPublicKeyInfo shall be used.

### 1553 7.1.2.2 Validation of UAID

1554 To be able to use the newly generated Device ID (UAID) and public key pair (DPKP), the device  
1555 Platform shall use the embedded private key (corresponding to manufacturer embedded public key  
1556 and certificate) to sign a token vouching for the fact that it (the Platform) has in fact generated the  
1557 DPKP and UAID and thus deferring the liability of the use of the DPKP to the new device owner.  
1558 This also allows the ecosystem to extend the trust from manufacturer certificate to a device issued  
1559 certificate for use in the new DPKP and UAID. The degree of trust is in dependent of the level of  
1560 hardening of the device SEE.

```
1561 Dev-Token=Info, Signature(hash(info))  
1562 Signature algorithm=ECDSA (can be same algorithm as that in EPC or that possible for  
1563 DPKP)  
1564 Hash algorithm=SHA256  
1565 Info=UAID | <Platform ID> | UAID_generation_data | validity  
1566 UAID_generation_data=data passed to the hash algorithm used to generate UAID.
```

1567 Validity=validity period in days (how long the token will be valid)

## 1568 **7.2 Device Ownership**

1569 This is an informative clause. Devices are logical entities that are security endpoints that have an  
1570 identity that is authenticable using cryptographic credentials. A Device is "un-owned" when it is  
1571 first initialized. Establishing device ownership is a process by which the device asserts its identity  
1572 to the DOTS and the DOTS provisions an owner identity. This exchange results in the device  
1573 changing its ownership state, thereby preventing a different DOTS from asserting administrative  
1574 control over the device.

1575 The ownership transfer process starts with the OBT discovering a new device that is "un-owned"  
1576 through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new device.  
1577 At the end of ownership transfer, the following is accomplished:

- 1578 1) The DOTS shall establish a secure session with new device.
- 1579 2) Optionally asserts any of the following:
  - 1580 a) Proximity (using PIN) of the OBT to the Platform.
  - 1581 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific  
1582 attributes.
- 1583 3) Determines the device identifier.
- 1584 4) Determines the device owner.
- 1585 5) Specifies the device owner (e.g. Device ID of the OBT).
- 1586 6) Provisions the device with owner's credentials.
- 1587 7) Sets the "Owned" state of the new device to TRUE.

1588 NOTE A Device which connects to the OCF Cloud still retains the ownership established at onboarding with the DOTS.

## 1589 **7.3 Device Ownership Transfer Methods**

### 1590 **7.3.1 OTM implementation requirements**

1591 This document provides specifications for several methods for ownership transfer. Implementation  
1592 of each individual ownership transfer method is considered optional. However, each device shall  
1593 implement at least one of the ownership transfer methods not including vendor specific methods.

1594 All OTMs included in this document are considered optional. Each vendor is required to choose  
1595 and implement at least one of the OTMs specified in this document. The OCF, does however,  
1596 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability  
1597 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with  
1598 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the preferred  
1599 approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in designing  
1600 vendor-specific OTMs.

1601 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer  
1602 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.  
1603 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the  
1604 /oic/sec/doxm Resource. The DOTS shall query the Device's supported credential types using the  
1605 credtypes Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision credentials  
1606 according to the credential types supported.

1607 Figure 13 depicts new Device discovery sequence.

### Discover New Devices Sequence

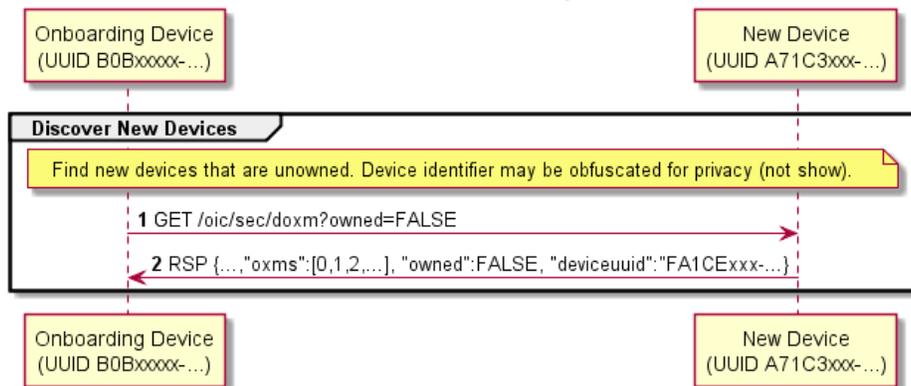


Figure 13 – Discover New Device Sequence

Table 1 – Discover New Device Details

Step	Description
1	The OBT queries to see if the new device is not yet owned.
2	The new device returns the "/oic/sec/doxm" Resource containing ownership status and supported OTMs. It also contains a temporal device ID that may change subsequent to successful owner transfer. The device should supply a temporal ID to facilitate discovery as a guest device. Clause 7.3.9 provides security considerations regarding selecting an OTM.

1612 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs  
 1613 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for  
 1614 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the  
 1615 new Device.

1616 The new device may have to perform some initialization steps at the beginning of an OTM. For  
 1617 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN  
 1618 value. The OBT shall POST to the oxmsel property of /oic/sec/doxm the value corresponding to the  
 1619 OTM being used, before performing other OTM steps. This POST notifies the new device that  
 1620 ownership transfer is starting.

1621 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT and  
 1622 the OBT to authenticate to the new device.

1623 The DOTS may perform additional provisioning steps subsequent to owner transfer success  
 1624 leveraging the established OTM session.

1625 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall  
1626 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array  
1627 includes a SVR.

### 1628 **7.3.2 SharedKey Credential Calculation**

1629 The SharedKey credential is derived using a PRF that accepts the key\_block value resulting from  
1630 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following  
1631 calculation to ensure interoperability across vendor products:

1632 SharedKey = PRF(Secret, Message);

1633 Where:

- 1634 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1635 - Secret is the key\_block resulting from the DTLS handshake
  - 1636 ▪ See IETF RFC 5246 Clause 6.3
  - 1637 ▪ The length of key\_block depends on cipher suite.
    - 1638 • (e.g. 96 bytes for TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
    - 1639 40 bytes for TLS\_PSK\_WITH\_AES\_128\_CCM\_8)
- 1640 - Message is a concatenation of the following:
  - 1641 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
    - 1642 • See "Clause 13.2.4 for specific DoxmTypes"
  - 1643 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
    - 1644 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
  - 1645 ▪ Device ID is new device's UUID Device ID
    - 1646 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1647 - SharedKey Length will be 32 octets.
  - 1648 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the leftmost 16 octets will be used.
  - 1649 DTLS sessions using 256 bit encryption cipher suites will use all 32 octets.

### 1650 **7.3.3 Certificate Credential Generation**

1651 The Certificate Credential will be used by Devices for secure bidirectional communication. The  
1652 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used  
1653 to mutually establish the authenticity of the Device. The onboarding details for certificate  
1654 generation will be specified in a later version of this document.

### 1655 **7.3.4 Just-Works OTM**

#### 1656 **7.3.4.1 Just-Works OTM General**

1657 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a  
1658 secure connection through which a device should be provisioned for use within the owner's OCF  
1659 Security Domain. Provisioning additional credentials and Resources is a typical step following  
1660 ownership establishment. The pre-shared key is called SharedKey.

1661 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite  
1662 defined for the Just-works OTM.

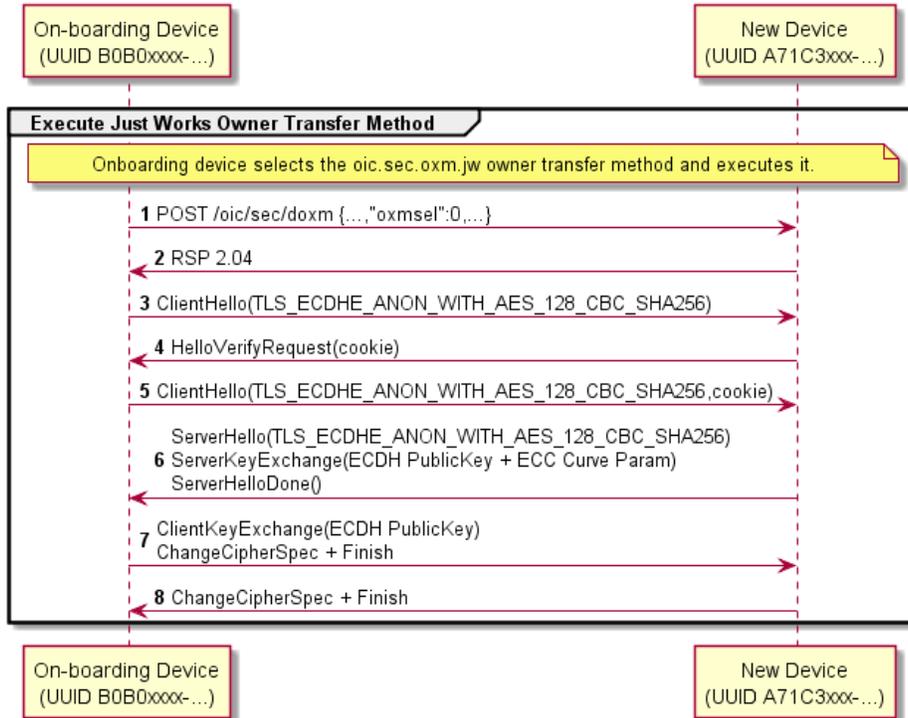
1663 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1664 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,  
1665 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

1666 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for  
1667 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1668 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.

**Perform Just-Works Owner Transfer Method**



**Figure 14 – A Just Works OTM**

**Table 2 – A Just Works OTM Details**

Step	Description
1, 2	The OBT notifies the Device that it selected the "Just Works" method.
3 - 8	A DTLS session is established using anonymous Diffie-Hellman. <sup>a</sup>

<sup>a</sup> This method assumes the operator is aware of the potential for man-in-the-middle attack and has taken precautions to perform the method in a clean-room network.

**7.3.4.2 Security Considerations**

Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this method presumes that both the OBT and the new device perform the "just-works" method assumes onboarding happens in a relatively safe environment absent of an attack device.

This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to the device.

1678 The new device should use a temporal device ID prior to transitioning to an owned device while it  
1679 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-  
1680 temporal device ID that could differ from the temporal value during the secure session in which  
1681 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not conflict  
1682 with a Device ID already in use. If it is already in use the existing credentials are used to establish  
1683 a secure session.

1684 An un-owned Device that also has established device credentials might be an indication of a  
1685 corrupted or compromised device.

### 1686 **7.3.5 Random PIN Based OTM**

#### 1687 **7.3.5.1 Random PIN OTM General**

1688 The Random PIN method establishes physical proximity between the new device and the OBT can  
1689 prevent man-in-the-middle attacks. The Device generates a random number that is communicated  
1690 to the OBT over an out-of-band channel. The definition of out-of-band communications channel is  
1691 outside the scope of the definition of device OTMs. The OBT and new Device use the PIN in a key  
1692 exchange as evidence that someone authorized the transfer of ownership by having physical  
1693 access to the new Device via the out-of-band-channel.

#### 1694 **7.3.5.2 Random PIN Owner Transfer Sequence**

1695 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.

### Perform Random PIN Device Owner Transfer Method

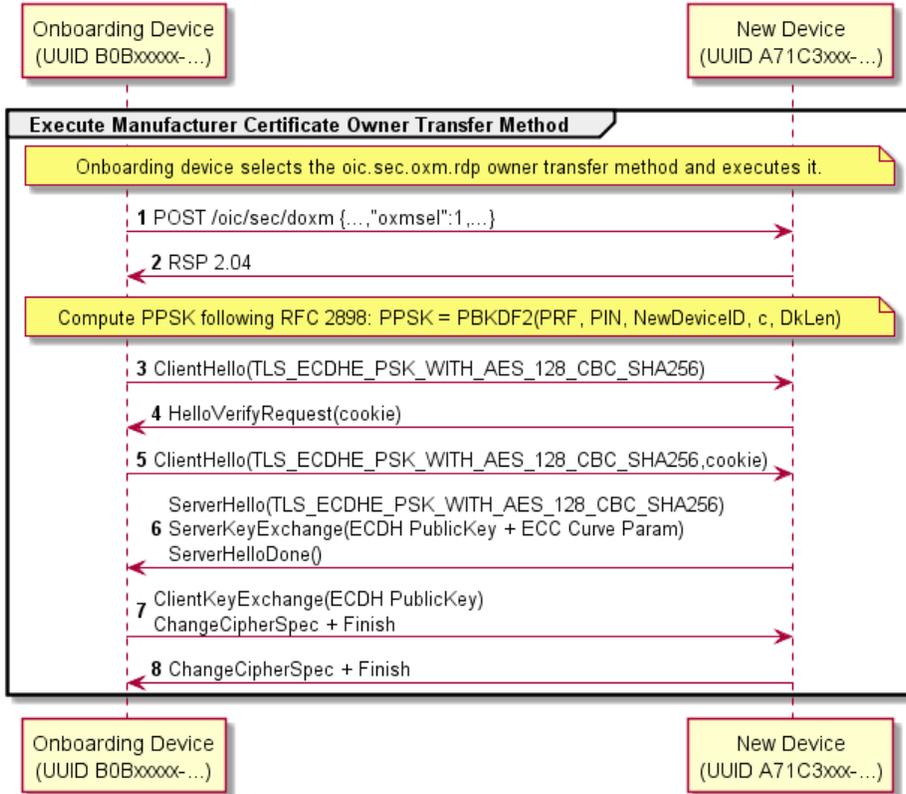


Figure 15 – Random PIN-based OTM

Table 3 – Random PIN-based OTM Details

Step	Description
1, 2	The OBT notifies the Device that it selected the "Random PIN" method.
3 - 8	A DTLS session is established using PSK-based Diffie-Hellman ciphersuite. The PIN is supplied as the PSK parameter. The PIN is randomly generated by the new device then communicated via an out-of-band channel that establishes proximal context between the new device and the OBT. The security principle is the attack device will be unable to intercept the PIN due to a lack of proximity.

1700 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF  
1701 RFC 2898 and a PIN exchanged via an out-of-band method to generate a pre-shared key. The PIN-  
1702 authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a PSK.

1703 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)

1704 The PBKDF2 function has the following parameters:

1705 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.

1706 - PIN – obtain via out-of-band channel.

1707 - Device ID – UUID of the new device.

1708 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2

1709 - c – Iteration count initialized to 1000

1710 - dkLen – Desired length of the derived PSK in octets.

### 1711 7.3.5.3 Security Considerations

1712 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with  
1713 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials  
1714 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,  
1715 allows an attacker to masquerade as the onboarded device.

1716 It is recommended that the entropy of the PIN be enough to withstand an online brute-force attack,  
1717 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-9a-z), or  
1718 a 7 character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack (MITM) is  
1719 when the attacker is active on the network and can intercept and modify messages between the  
1720 OBT and device. In the MITM attack, the attacker must recover the PIN from the key exchange  
1721 messages in "real time", i.e., before the peers time out and abort the connection attempt. Having  
1722 recovered the PIN, he can complete the authentication step of key exchange. The guidance given  
1723 here calls for a minimum of 40 bits of entropy, however, the assurance this provides depends on  
1724 the resources available to the attacker. Given the parallelizable nature of a brute force guessing  
1725 attack, the attack enjoys a linear speedup as more cores/threads are added. A more conservative  
1726 amount of entropy would be 64 bits. Since the Random PIN OTM requires using a DTLS ciphersuite  
1727 that includes an ECDHE key exchange, the security of the Random PIN OTM is always at least  
1728 equivalent to the security of the JustWorks OTM.

1729 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN. The  
1730 rationale is to increase the cost of a brute force attack, by increasing the cost of each guess in the  
1731 attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an effective way  
1732 to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify the reduction,  
1733 since an X-fold increase in time spent by the honest peers does not directly translate to an X-fold  
1734 increase in time by the attacker. This asymmetry is because the attacker may use specialized  
1735 implementations and hardware not available to honest peers. For this reason, when deciding how  
1736 much entropy to use for a PIN, it is recommended that implementers assume PBKDF2 provides no  
1737 security, and ensure the PIN has sufficient entropy.

1738 The Random PIN device OTM security depends on an assumption that a secure out-of-band  
1739 method for communicating a randomly generated PIN from the new device to the OBT exists. If the  
1740 OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN, and  
1741 the attacks described above apply. The out-of-band mechanism should be chosen such that it  
1742 requires proximity between the OBT and the new device. The attacker is assumed to not have  
1743 compromised the out-of-band-channel. As an example OOB channel, the device may display a PIN  
1744 to be entered into the OBT software. Another example is for the device to encode the PIN as a 2D  
1745 barcode and display it for a camera on the OBT device to capture and decode.

1746 **7.3.6 Manufacturer Certificate Based OTM**

1747 **7.3.6.1 Manufacturer Certificate Based OTM General**

1748 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the  
1749 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device  
1750 and the OBT.

1751 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust  
1752 anchor.

1753 For some environments, policies or administrators, additional information about device  
1754 characteristics may be sought. This list of additional attestations that OCF may or may not have  
1755 tested (understanding that some attestations are incapable of testing or for which testing may be  
1756 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3  
1757 extension described in 9.4.2.2.6.

1758 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with  
1759 certificate data to authenticate their identities with the OBT in the process of bringing a new device  
1760 into operation on an OCF Security Domain. The onboarding process involves several discrete steps:

- 1761 1) Pre-on-board conditions
- 1762 a) The credential element of the Device's credential Resource (/oic/sec/cred) containing the  
1763 manufacturer certificate shall be identified by the properties:
    - 1764 i) the subject Property shall refer to the Device
    - 1765 ii) the credusage Property shall contain the string "oic.sec.cred.mfgcert" to indicate that  
1766 the credential contains a manufacturer certificate
  - 1767 b) The manufacturer certificate chain shall be contained in the identified credential element's  
1768 publicdata Property.
  - 1769 c) The device shall contain a unique and immutable ECC asymmetric key pair.
  - 1770 d) If the device requires authentication of the OBT as part of ownership transfer, it is presumed  
1771 that the OBT has been registered and has obtained a certificate for its unique and immutable  
1772 ECC asymmetric key pair signed by the predetermined Trust Anchor.
  - 1773 e) User has configured the OBT app with network access info and account info (if any).
- 1774 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the  
1775 Device may authenticate the OBT to verify the OBT signature.
- 1776 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready  
1777 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted link  
1778 in accordance with the negotiated cipher suite.

1779 **7.3.6.2 Certificate Profiles**

1780 See 9.4.2 for details.

1781 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

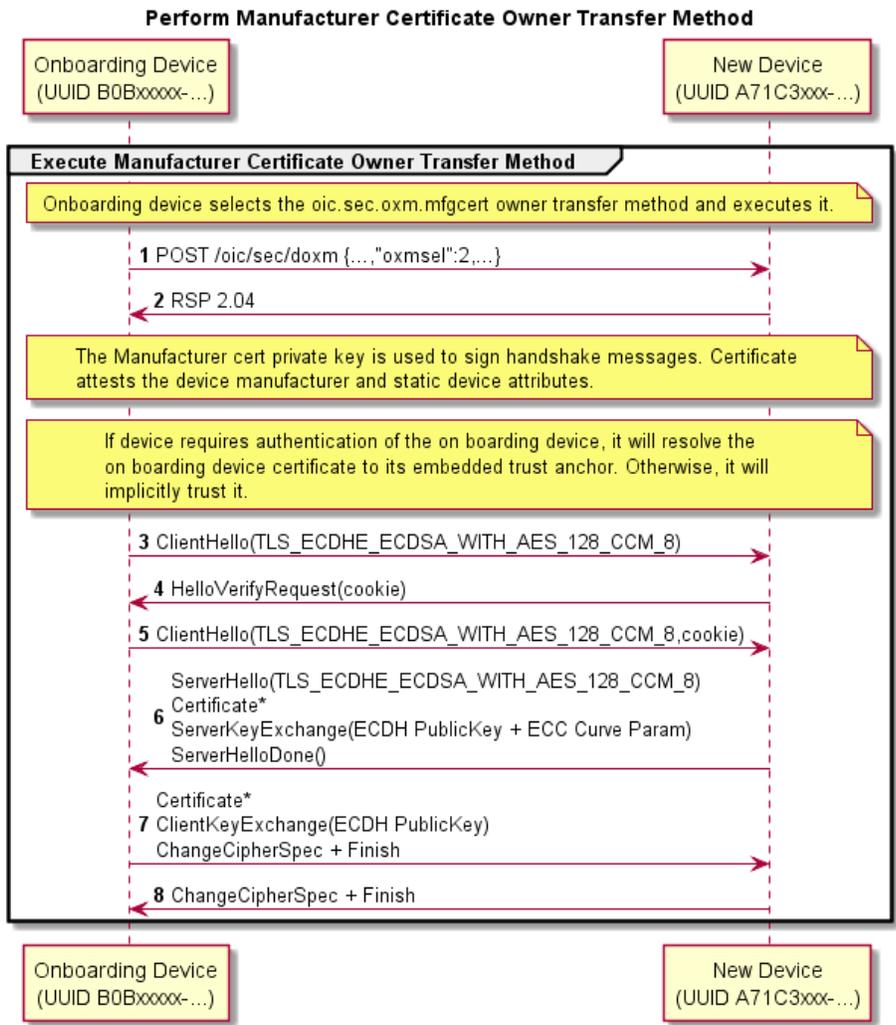
1782 In order for full, mutual authentication to occur between the device and the OBT, both the device  
1783 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This implies  
1784 that OCF may need to obtain services from a Certificate Authority (e.g. Symantec, Verisign, etc.)  
1785 to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors are derived.

1786 The OBT shall authenticate the device during onboarding. However, the device is not required to  
1787 authenticate the OBT due to potential resource constraints on the device.

1788 In the case where the Device does NOT authenticate the OBT software, there is the possibility of  
 1789 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,  
 1790 which can compromise OCF Security Domain access credentials and/or personal information.

1791 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1792 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.



1793

1794

**Figure 16 – Manufacturer Certificate Based OTM Sequence**

1795

1796

**Table 4 – Manufacturer Certificate Based OTM Details**

Step	Description
1, 2	The OBT notifies the Device that it selected the "Manufacturer Certificate" method.
3 - 8	A DTLS session is established using the device's manufacturer certificate and optional OBT certificate. The device's manufacturer certificate may contain data attesting to the Device hardening and security properties.

1797 **7.3.6.5 Security Considerations**

1798 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of  
1799 assurance that the private key cannot be compromised.

1800 The Platform manufacturer issues the manufacturer certificate and attests the private key  
1801 protection mechanism.

1802 **7.3.7 Vendor Specific OTMs**

1803 **7.3.7.1 Vendor Specific OTM General**

1804 The OCF anticipates situations where a vendor will need to implement an OTM that accommodates  
1805 manufacturing or Device constraints. The Device OTM resource is extensible for this purpose.  
1806 Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

1807 – The OBT must determine which credential types are supported by the Device. This is  
1808 accomplished by querying the Device's /oic/sec/doxm Resource to identify supported credential  
1809 types.

1810 – The OBT provisions the Device with OC(s).

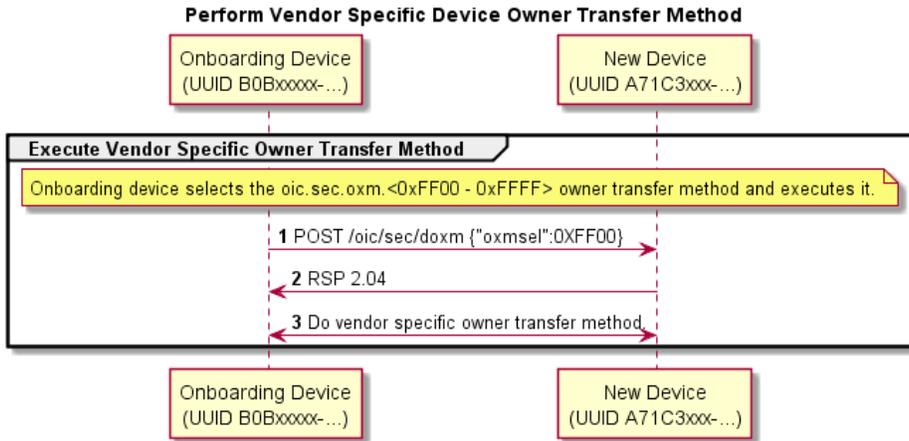
1811 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.

1812 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security  
1813 Domain subsequent to ownership establishment.

1814 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks  
1815 to be performed by a security service.

1816 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1817 Random PIN-based OTM sequence example is shown in Figure 17 and steps described in Table 5.



**Figure 17 – Vendor-specific Owner Transfer Sequence**

**Table 5 – Vendor-specific Owner Transfer Details**

Step	Description
1, 2	The OBT selects a vendor-specific OTM.
3	The vendor-specific OTM is applied

**7.3.7.3 Security Considerations**

The vendor is responsible for considering security threats and mitigation strategies.

**7.3.8 Establishing Owner Credentials**

Once the OBT and the new Device have authenticated and established an encrypted connection using one of the defined OTM methods.

Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security Domain access information, provisioning functions, shared keys, or Kerberos tickets.

The OBT might then provision the new Device with additional credentials for Device management and Device-to-Device communications. These credentials may consist of certificates with signatures, UAID based on the Device public key, PSK, etc.

The steps for establishing Device's owner credentials (OC) are:

- 1) The OBT shall establish the Device ID and Device owner uuid - Figure 18 and Table 6
- 2) The OBT then establishes Device's OC - Figure 19 and Table 7. This can be either:
  - a) Symmetric credential - Figure 20 and Table 8.
  - b) Asymmetric credential - Figure 21 and Table 9.
- 3) Configure Device services - Figure 22 and Table 10.
- 4) Configure Device for peer to peer interaction - Figure 23 and Table 11.

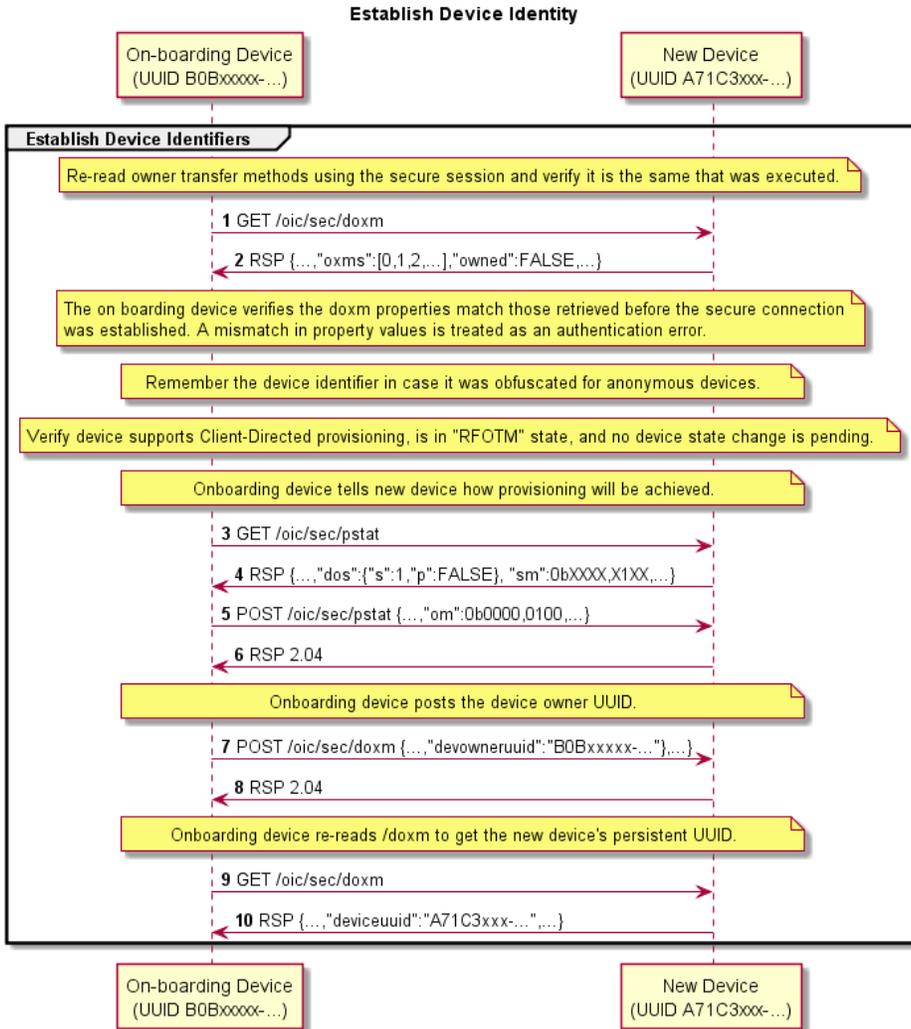


Figure 18 – Establish Device Identity Flow

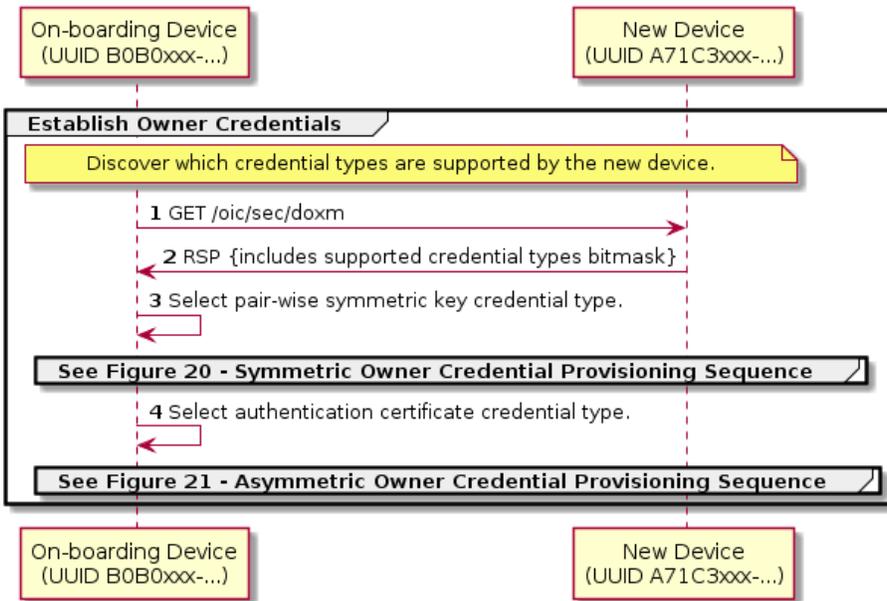
1842

1843

Table 6 – Establish Device Identity Details

Step	Description
1, 2	The OBT obtains the doxm properties again, using the secure session. It verifies that these properties match those retrieved before the authenticated connection. A mismatch in parameters is treated as an authentication error.
3, 4	The OBT queries to determine if the Device is operationally ready to transfer Device ownership.
5, 6	The OBT asserts that it will follow the Client provisioning convention.
7, 8	The OBT asserts itself as the owner of the new Device by setting the Device ID to its ID.
9, 10	The OBT obtains doxm properties again, this time Device returns new Device persistent UUID.

Establish Owner Credentials Sequence



1844

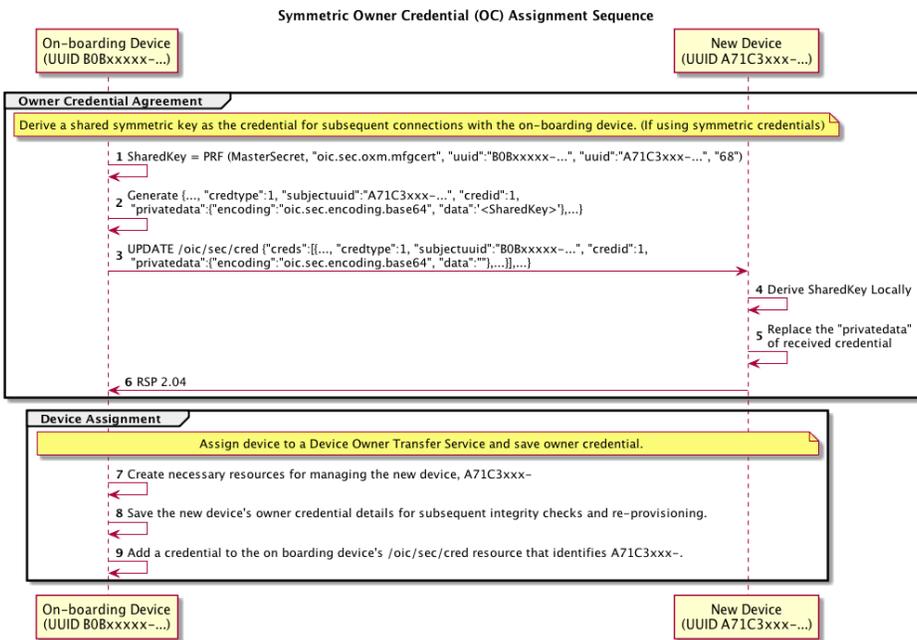
1845

Figure 19 – Owner Credential Selection Provisioning Sequence

1846  
1847

**Table 7 – Owner Credential Selection Details**

Step	Description
1, 2	The OBT obtains the doxm properties to check ownership transfer mechanism supported on the new Device.
3, 4	The OBT uses selected credential type for ownership provisioning.



1848  
1849  
1850  
1851

**Figure 20 – Symmetric Owner Credential Provisioning Sequence**

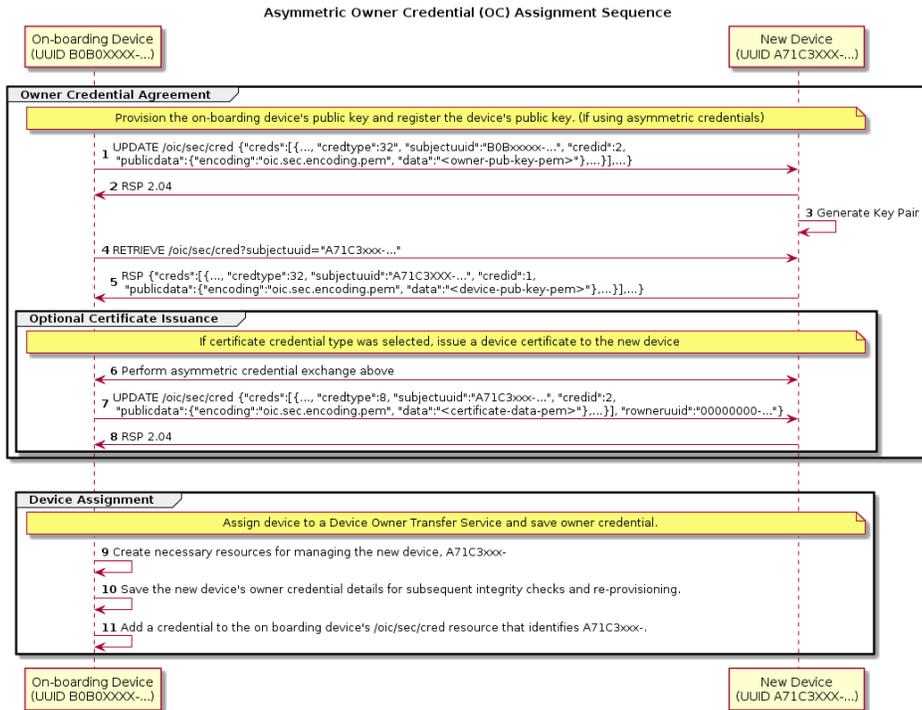
**Table 8 – Symmetric Owner Credential Assignment Details**

Step	Description
1, 2	The OBT uses a pseudo-random-function (PRF), the master secret resulting from the DTLS handshake, and other information to generate a symmetric key credential resource Property - SharedKey.
3	The OBT creates a credential resource Property set based on SharedKey and then sends the resource Property set to the new Device with empty "privatedata" Property value.

4, 5	The new Device locally generates the SharedKey and updates it to the "privatedata" Property of the credential resource Property set.
6	The new Device sends a success message.
7	The onboarding service creates a subjects resource for the new device (e.g./A71C3xxx-...)
8	The onboarding service provisions its /oic/svc/dots/subjects/A71C3xxx-/cred resource with the owner credential. Credential type is SYMMETRIC KEY.
9	(optional) The onboarding service provisions it's own "/oic/sec/cred" resource with the owner credential for new device. Credential type is SYMMETRIC KEY.

1852 In particular, if the OBT selects symmetric owner credentials:

- 1853 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
- 1854 described in 7.3.2.
- 1855 – The OBT shall send an empty key to the new Device's /oic/sec/cred Resource, identified as a
- 1856 symmetric pair-wise key.
- 1857 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
- 1858 generate the Shared Key using the SharedKey Credential Calculation method described in 7.3.2
- 1859 and store it with the owner credential.
- 1860 – The new Device shall use the Shared Key owner credential(s) stored via the /oic/sec/cred
- 1861 Resource to authenticate the owner during subsequent connections.



1862  
1863  
1864  
1865  
**Figure 21 – Asymmetric Owner Credential Provisioning Sequence**

**Table 9 – Asymmetric Owner Credential Assignment Details**

Step	Description
If an asymmetric or certificate owner credential type was selected by the OBT	
1, 2	The OBT creates an asymmetric type credential Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric type credential Resource Property set generated at step 25. It may be used subsequently to authenticate the new Device.
If certificate owner credential type is selected by the OBT	
6-8	The steps for creating an asymmetric credential type are performed. In addition, the OBT instantiates a newly-created certificate (or certificate chain) on the new Device.

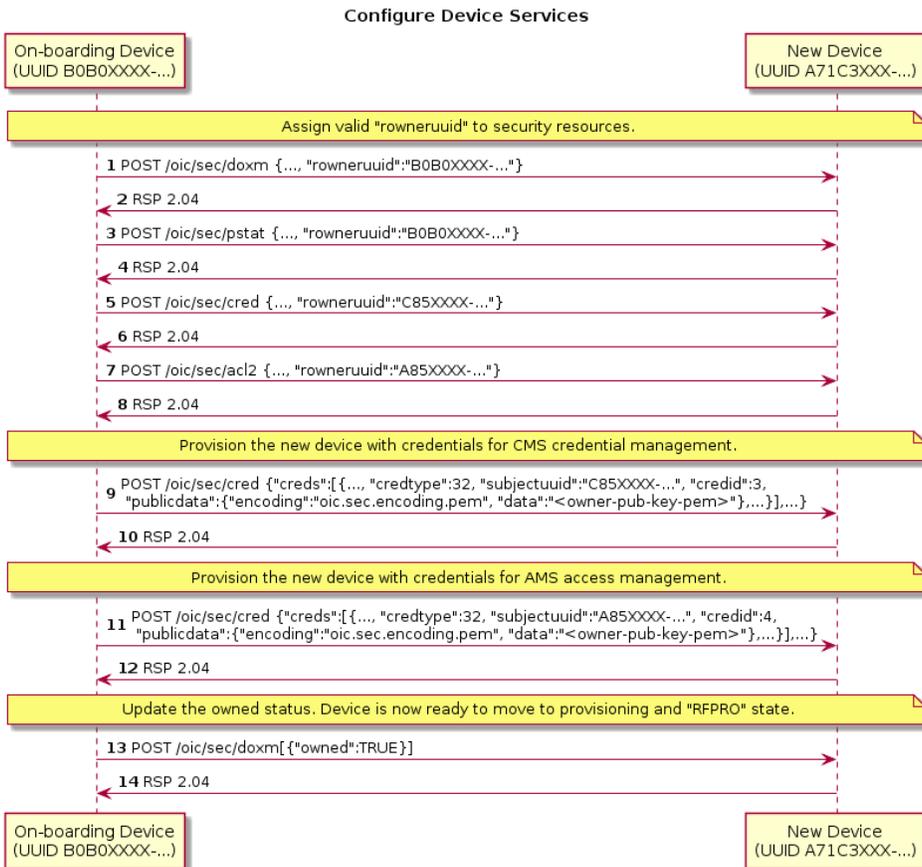
9	The onboarding service creates a subjects resource for the new device (e.g./A71C3xxx-...)
10	The onboarding service provisions its /oic/svc/dots/subjects/A71C3xxx-/cred resource with the owner credential. Credential type is PUBLIC KEY.
11	(optional) The onboarding service provisions it's own /oic/sec/cred resource with the owner credential for new device. Credential type is PUBLIC KEY.
12	(optional) The onboarding service provisions it's own /oic/sec/cred resource with the owner credential for new device. Credential type is CERTIFICATE.

1866 If the OBT selects asymmetric owner credentials:

- 1867 – The OBT shall add its public key to the new Device's /oic/sec/cred Resource, identified as an
- 1868 Asymmetric Encryption Key.
- 1869 – The OBT shall query the /oic/sec/cred Resource from the new Device, supplying the new
- 1870 Device's UUID via the SubjectID query parameter. In response, the new Device shall return the
- 1871 public Asymmetric Encryption Key, which the OBT shall retain for future owner authentication
- 1872 of the new Device.

1873 If the OBT selects certificate owner credentials:

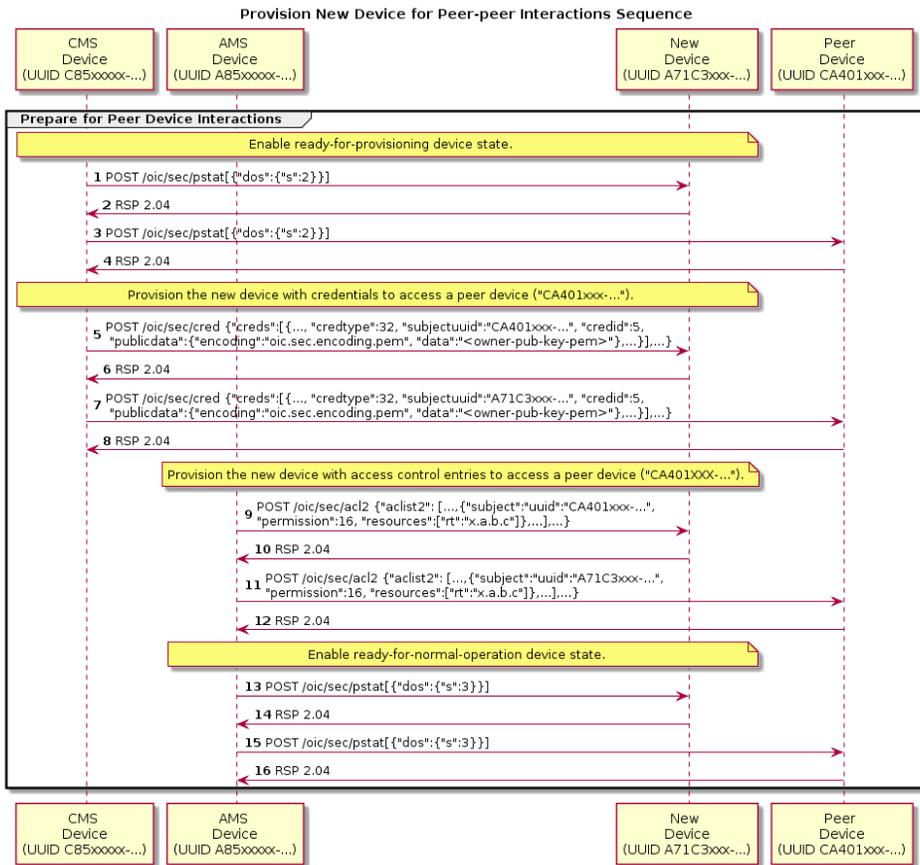
- 1874 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the
- 1875 public key returned by the new Device, signed by a mutually-trusted CA, and complying with
- 1876 the Certificate Credential Generation requirements defined in 7.3.3.
- 1877 – The OBT shall add the newly-created certificate chain to the /oic/sec/cred Resource, identified
- 1878 as an Asymmetric Signing Key with Certificate.



**Figure 22 – Configure Device Services**

**Table 10 – Configure Device Services Detail**

Step	Description
1 - 8	The OBT assigns rowneruuid for different SVRs.
9 - 10	Provision the new Device with credentials for CMS
11 - 12	Provision the new Device with credentials for AMS
13 - 14	Update the oic.sec.doxm.owned to TRUE. Device is ready to move to provision and RFPRO state.



1883

1884

1885

**Figure 23 – Provision New Device for Peer to Peer Interaction Sequence**

**Table 11 – Provision New Device for Peer to Peer Details**

Step	Description
1 - 4	The OBT set the Devices in the ready for provisioning status by setting oic.sec.pstat.dos to 2.
5 - 8	The OBT provision the Device with peer credentials
9 - 12	The OBT provision the Device with access control entities for peer Devices.
13 - 16	Enable Device to RFNOP state by setting oic.sec.pstat.dos to 3.

1886 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1887 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are  
1888 acceptable when transferring ownership of a new Device. Some of the factors to be considered  
1889 when determining those requirements are:

- 1890 – The security considerations described for each of the OTMs
- 1891 – The probability that a man-in-the-middle attacker might be present in the environment used to  
1892 perform the Ownership Transfer

1893 For example, the operator of an OBT might require that all of the Devices being onboarded support  
1894 either the Random PIN or the Manufacturer Certificate OTM.

1895 When such a local OTM policy exists, the OBT should try to use just the OTMs that are acceptable  
1896 according to that policy, regardless of the doxm contents obtained during step 1 from the sequence  
1897 diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an unauthenticated and/or  
1898 unencrypted connection between the OBT and the Device, the contents of the response to the GET  
1899 request might have been tampered by a man-in-the-middle attacker. For example, the list of OTMs  
1900 supported by the new Device might have been altered by the attacker.

1901 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new  
1902 Device to fail. In such cases, the OBT has no way of determining if the session failed because the  
1903 new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle injected  
1904 such a failure into the communication between the OBT and the new Device.

1905 The current version of this document leaves the design and user experience related to the OTM  
1906 policy as OBT implementation details.

1907 **7.3.10 Security Profile Assignment**

1908 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results  
1909 could be accessed from a manufacturer's certificate, OCF web server or other public repository.  
1910 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device is  
1911 authorized to possess and configures the Device with the subset of evaluated security profiles best  
1912 suited for the OCF Security Domain owner's intended segmentation strategy.

1913 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles" Property  
1914 of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification process.  
1915 The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the values  
1916 contained in the "supportedprofiles". The manufacturer default value shall be re-asserted when the  
1917 Device transitions to RESET Device State.

1918 The OCF Device shall only allow the /oic/sec/sp Resource to be updated when the Device is in one  
1919 of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as  
1920 directed by a Security Profile.

1921 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a subset  
1922 of the OCF Security Profiles values the Device achieved as part of OCF Conformance testing. The  
1923 DOTS may locate conformance results by inspecting manufacturer certificates supplied with the  
1924 OCF Device by selecting the "credusage" Property of the /oic/sec/cred Resource having the value  
1925 of "oic.sec.cred.mfgcert". The DOTS may further locate conformance results by visiting a well-  
1926 known OCF web site URI corresponding to the ocCPLAttributes extension fields (clause 9.4.2.2.7).  
1927 The DOTS may select a subset of Security Profiles (from those evaluated by OCF conformance  
1928 testing) based on a local policy.

1929 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries to  
1930 allow DOTS access subsequent to onboarding.

1931 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the value  
1932 that most correctly depicts the OCF Security Domain owner's intended Device deployment strategy.

1933 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID") to  
1934 indicate the OCF Security Domain owner's intention to segment the OCF Security Domain  
1935 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the  
1936 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security  
1937 Profiles when issuing role credentials.

1938 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control  
1939 entries that include the role designation(s).

## 1940 **7.4 Provisioning**

### 1941 **7.4.1 Provisioning Flows**

#### 1942 **7.4.1.1 Provisioning Flows General**

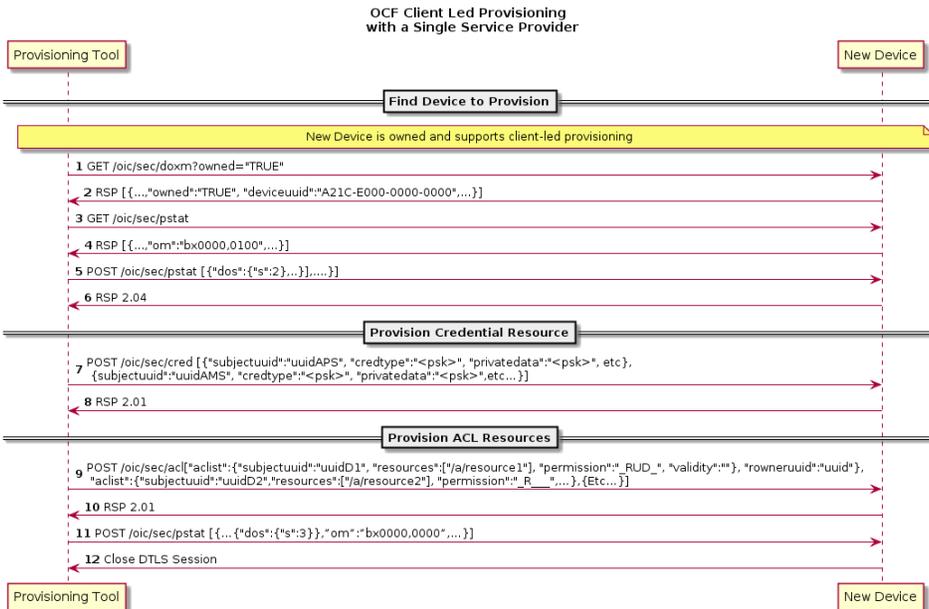
1943 As part of onboarding a new Device a secure channel is formed between the new Device and the  
1944 OBT. Subsequent to the Device ownership status being changed to "owned", there is an opportunity  
1945 to begin provisioning. The OBT decides how the new Device will be managed going forward and  
1946 provisions the support services that should be subsequently used to complete Device provisioning  
1947 and on-going Device management.

1948 The Device employs a Server-directed or Client-directed provisioning strategy. The /oic/sec/pstat  
1949 Resource identifies the provisioning strategy and current provisioning status. The provisioning  
1950 service should determine which provisioning strategy is most appropriate for the OCF Security  
1951 Domain. See 13.8 for additional detail.

#### 1952 **7.4.1.2 Client-directed Provisioning**

1953 Client-directed provisioning relies on a provisioning service that identifies Servers in need of  
1954 provisioning then performs all necessary provisioning duties.

1955 An example of Client-directed provisioning is shown in Figure 24 and steps described in Table 12.



1956  
1957  
1958

**Figure 24 – Example of Client-directed provisioning**

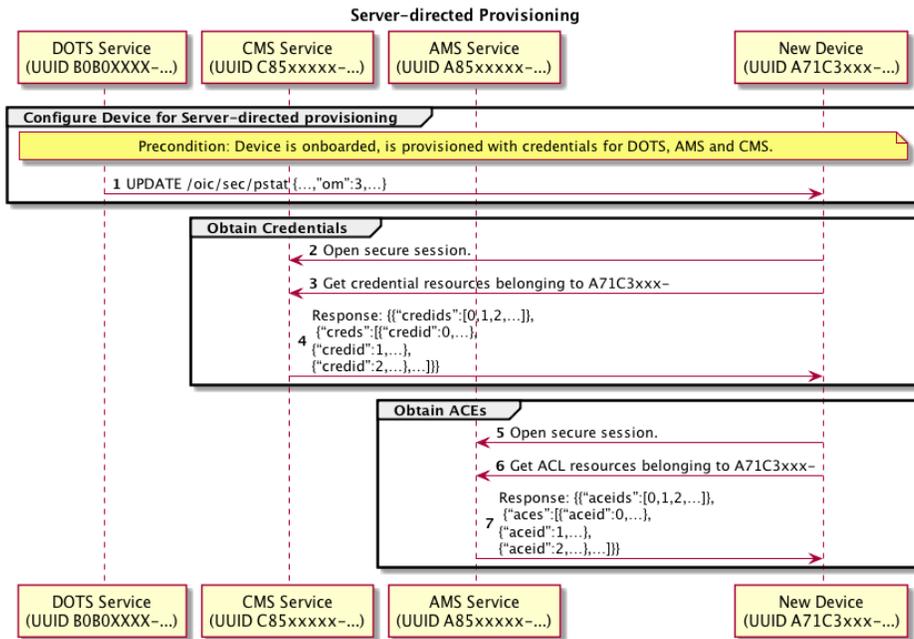
**Table 12 – Steps describing Client -directed provisioning**

Step	Description
1	Discover Devices that are owned and support Client-directed provisioning.
2	The "/oic/sec/doxm" Resource identifies the Device and it's owned status.
3	Provisioning Tool (PT) obtains the new Device's provisioning status found in /oic/sec/pstat Resource
4	The pstat Resource describes the types of provisioning modes supported and which is currently configured. A Device manufacturer should set a default current operational mode (om). If the Om isn't configured for Client-directed provisioning, its om value can be changed.
5 - 6	Change Device state to Ready-for-Provisioning.
7 - 8	PT instantiates the /oic/sec/cred Resource. It contains credentials for the provisioned services and other Devices
9 - 10	PT instantiates "/oic/sec/acl2" Resources.
11	The new Device provisioning status mode is updated to reflect that ACLs have been configured. (Ready-for-Normal-Operation state)
12	The secure session is closed.

1959 **7.4.1.3 Server-directed Provisioning**

1960 Server-directed provisioning relies on the Server (i.e. New Device) for directing much of the  
 1961 provisioning work. As part of the onboarding process the support services used by the Server to  
 1962 seek additional provisioning are provisioned. The New Device uses a self-directed, state-driven  
 1963 approach to analyze current provisioning state, and tries to drive toward target state. This example  
 1964 assumes a single support service is used to provision the new Device.

1965 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1966 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**  
 1967

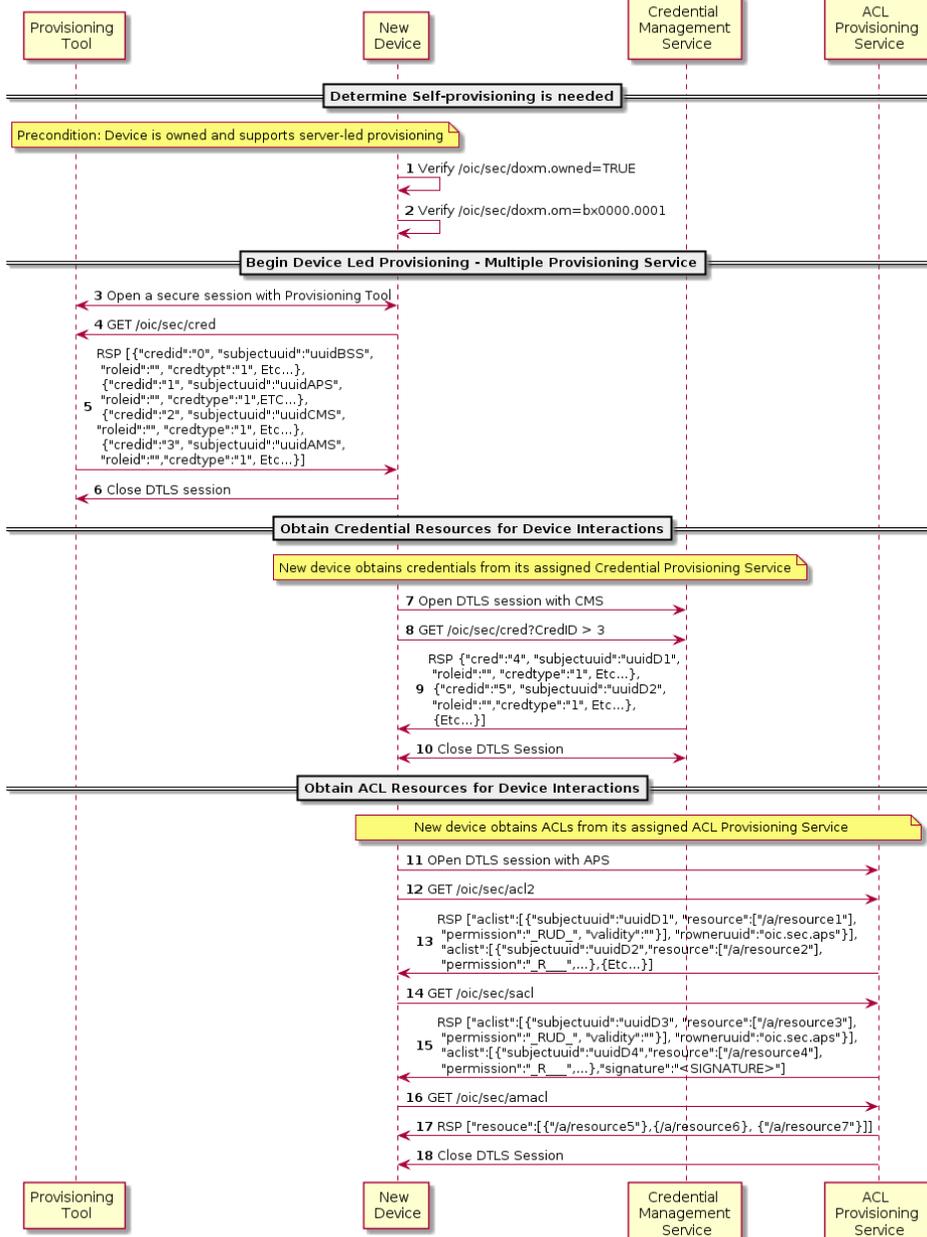
1968 **Table 13 – Steps for Server-directed provisioning using a single provisioning service**

Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a secure session with the provisioning tool using the <code>/oic/sec/doxm</code> . DevOwner value to open a TLS connection using SharedKey.

8 – 9	The new Device gets the /oic/sec/cred Resources. It contains credentials for the provisioned services and other Devices.
11 – 12	The new Device gets the /oic/sec/acl2 Resources.
14	The secure session is closed.

1969 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**  
1970 A Server-directed provisioning flow, involving multiple support services distributes the provisioning  
1971 work across multiple support services. Employing multiple support services is an effective way to  
1972 distribute provisioning workload or to deploy specialized support. The example in Figure 26  
1973 demonstrates using a provisioning tool to configure two support services, a CMS and an AMS.  
1974 Steps for the example are described in Table 14.

OCF Server Led Provisioning with Multiple Service Providers



1976 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1977 **Table 14 – Steps for Server-directed provisioning involving multiple support services**

Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device initiates a secure session with the provisioning tool using the /oic/sec/doxm. DevOwner value to open a TLS connection using SharedKey.
4-5	The new Device gets credentials Resource for the provisioned services and other Devices
6	The new Device closes the DTLS session with the provisioning tool.
7	The new Device finds the CMS from the /oic/sec/cred Resource, rowneruid Property and opens a DTLS connection. The new device finds the credential to use from the /oic/sec/cred Resource.
8-9	The new Device requests additional credentials that are needed for interaction with other devices.
10	The DTLS connection is closed.
11	The new Device finds the ACL provisioning and management service from the /oic/sec/acl2 Resource, rowneruid Property and opens a DTLS connection. The new device finds the ACL to use from the /oic/sec/acl2 Resource.
12-13	The new Device gets ACL Resources that it will use to enforce access to local Resources.
14-15	The new Device should get SACL Resources immediately or in response to a subsequent Device Resource request.
16-17	The new Device should also get a list of Resources that should consult an Access Manager for making the access control decision.
18	The DTLS connection is closed.

1978 **7.5 Device Provisioning for OCF Cloud**

1979 **7.5.1 Cloud Provisioning General**

1980 The Device that connects to the OCF Cloud shall support the oic.r.coapcloudconf Resource on  
 1981 Device and following SVRs on the OCF Cloud: "/oic/sec/account", "/oic/sec/session",  
 1982 "/oic/sec/tokenrefresh".

1983 The OCF Cloud is expected to use a secure mechanism for associating a Mediator with an OCF  
 1984 Cloud User. The choice of mechanism is up to the OCF Cloud. Example, mechanisms include HTTP  
 1985 authentication (with username and password) or OAuth 2.0 (using an Authorization Server which  
 1986 could be operated by the OCF Cloud provider or a third party). OCF Cloud is expected to ensure  
 1987 that the suitable authentication mechanism is used to authenticate the OCF Cloud User.

1988 **7.5.2 Device Provisioning by Mediator**

1989 The Mediator and the Device shall use the secure session to provision the Device to connect with  
 1990 the OCF Cloud.

1991 The Mediator obtains an Access Token from the OCF Cloud as described in ISO/IEC 30118-8:2018.  
 1992 This Access Token is then used by the Device for registering with the OCF Cloud as described in

1993 10.5. The OCF Cloud maintains a map where Access Token and Mediator provided Device ID are  
 1994 stored. At the time of Device Registration OCF Cloud validates the Access Token and associates  
 1995 the TLS session with corresponding Device ID.

1996 The Mediator provisions the Device, as described in ISO/IEC 30118-8:2018. The Mediator  
 1997 provisions OCF Cloud URI to the "cis" Property of "oic.r.coapcloudconf" Resource, OCF Cloud  
 1998 UUID to the "sid" Property of "oic.r.coapcloudconf" Resource and per-device Access Token to the  
 1999 "at" Property of "oic.r.coapcloudconf" Resource on Device. Provisioned "at" is to be treated by  
 2000 Device as an Access Token with "Bearer" token type as defined in IETF RFC 6750.

2001 For the purposes of access control, the Device shall identify the OCF Cloud using the OCF Cloud  
 2002 UUID in the Common Name field of the End-Entity certificate used to authenticate the OCF Cloud.

2003 AMS should configure the ACE2 entries on a Device so that the Mediator(s) is the only Device(s)  
 2004 with UPDATE permission for the oic.r.coapcloudconf Resource.

2005 The AMS should configure the ACE2 entries on the Device to allow request from the OCF Cloud.  
 2006 By request from the Mediator, the AMS removes old ACL2 entries with previous OCF Cloud UUID.  
 2007 This request happens before "oic.r.coapcloudconf" is configured by the Mediator for the new OCF  
 2008 Cloud. The Mediator also requests AMS to set the OCF Cloud UUID as the "subject" Property for  
 2009 the new ACL2 entries. AMS may use "sid" Property of "oic.r.coapcloudconf" Resource as the  
 2010 current OCF Cloud UUID. AMS could either provision a wildcard entry for the OCF Cloud or  
 2011 provision an entry listing each Resource published on the Device.

2012 If OCF Cloud provides "redirecturi" Value as response during Device Registration, the redirected-  
 2013 to OCF Cloud is assumed to have the same OCF Cloud UUID and to use the same trust anchor.  
 2014 Otherwise, presented OCF Cloud UUID wouldn't match the provisioned ACL2 entries.

2015 The Mediator should provision the oic.r.coapcloudconf Resource with the Properties in Table 15.  
 2016 These details once provisioned are used by the Device to perform Device Registration to the OCF  
 2017 Cloud. After the initial registration, the Device should use updated values received from the OCF  
 2018 Cloud instead. If OCF Cloud User wants the Device to re-register with the OCF Cloud, they can  
 2019 use the Mediator to re-provision the oic.r.coapcloudconf Resource with the new values.

2020 **Table 15 – Mapping of Properties of the oic.r.account and oic.r.coapcloudconf Resources**

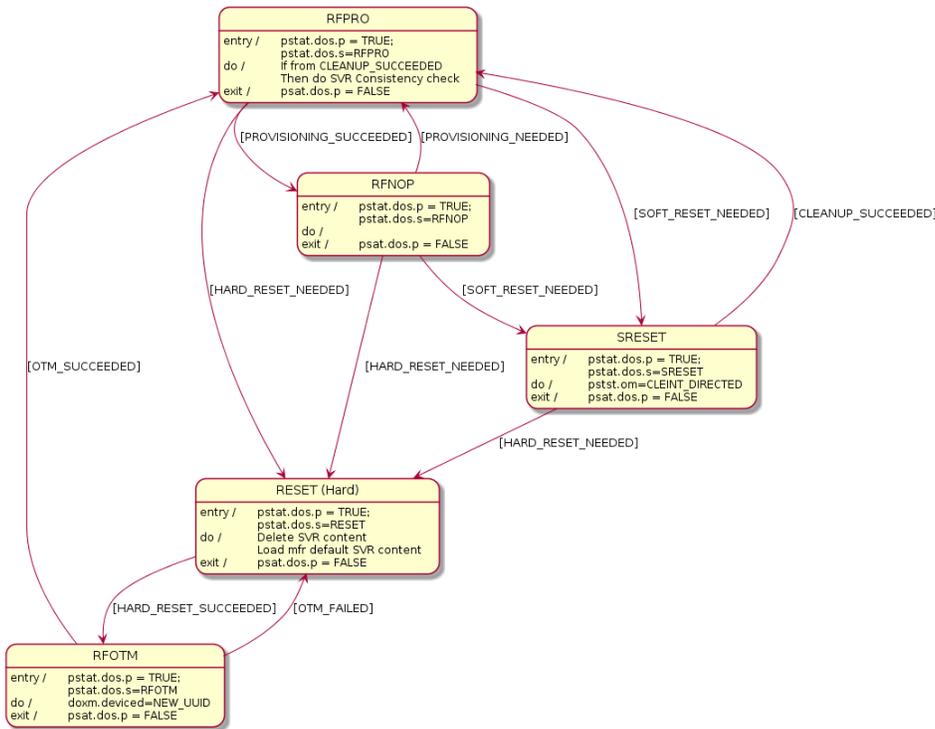
Property Name	oic.r.coapcloudconf	oic.r.account	Description
Authorization Provider Name	apn	authprovider	The Authorization Provider through which Access Token was obtained.
OCF Cloud URL	cis	-	This is the URL connection is established between Device and OCF Cloud.
Access Token	at	accesstoken	The unique token valid only for the Device.
OCF Cloud UUID	sid	-	This is the identity of the OCF Cloud that the Device is configured to use.

2021 **8 Device Onboarding State Definitions**

2022 **8.1 Device Onboarding General**

2023 As explained in 5.3, the process of onboarding completes after the ownership of the Device has  
 2024 been transferred and the Device has been provisioned with relevant configuration/services as  
 2025 explained in 5.4.. The Figure 27 shows the various states a Device can be in during the Device  
 2026 lifecycle.

2027 The /pstat.dos.s Property is RW by the /oic/sec/pstat resource owner (e.g. "doxs" service) so that  
 2028 the resource owner can remotely update the Device state. When the Device is in RFNOP or RFPRO,  
 2029 ACLs can be used to allow remote control of Device state by other Devices. When the Device state  
 2030 is SRESET the Device OC may be the only indication of authorization to access the Device. The  
 2031 Device owner may perform low-level consistency checks and re-provisioning to get the Device  
 2032 suitable for a transition to RFPRO.



2033 **Figure 27 – Device state model**  
 2034

2035 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the "Ready  
 2036 for Normal Operation" state where it has all it needs in order to start interoperating with other  
 2037 Devices. 8.2 specifies the minimum mandatory configuration that a Device shall hold in order to be  
 2038 considered as "Ready for Normal Operation".

2039 In the event of power loss or Device failure, the Device should remain in the same state that it was  
 2040 in prior to the power loss / failure

2041 If a Device or resource owner OBSERVEs /pstat.dos.s, then transitions to SRESET will give early  
 2042 warning notification of Devices that may require SVR consistency checking.

2043 In order for onboarding to function, the Device shall have the following Resources installed:

- 2044 1) "/oic/sec/doxm" Resource
- 2045 2) "/oic/sec/pstat" Resource

2046 3) "/oic/sec/cred" Resource

2047 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5  
2048 and 8.6.

## 2049 **8.2 Device Onboarding-Reset State Definition**

2050 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset  
2051 also defines a state where the Device asset is ready to be transferred to another party.

2052 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces Platform  
2053 reset. All Devices hosted on the same Platform transition their Device states to RESET when the  
2054 Platform reset is asserted.

2055 The following Resources and their specific properties shall have the value as specified:

- 2056 1) The owned Property of the "/oic/sec/doxm" Resource shall transition to FALSE.
- 2057 2) The devowneruid Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2058 3) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2059 implemented.
- 2060 4) The deviceuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2061 default value.
- 2062 5) The deviceid Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2063 default value, if this Property is implemented.
- 2064 6) The sct Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's default  
2065 value.
- 2066 7) The oxmsel Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2067 default value.
- 2068 8) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2069 9) The dos Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RESET"  
2070 state and dos.p shall equal "FALSE".
- 2071 10)
- 2072 11) The om (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the  
2073 manufacturer default value.
- 2074 12) The sm (supported operational modes) Property of the /oic/sec/pstat Resource shall be set to  
2075 the manufacturer default value.
- 2076 13) The rowneruid Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl2", "/oic/sec/amacl",  
2077 "/oic/sec/sacl", and "/oic/sec/cred" Resources shall be nil UUID.
- 2078 14) The supportedprofiles Property of the /oic/sec/sp Resource shall be set to the manufacturer  
2079 default value.
- 2080 15) The currentprofile Property of the /oic/sec/sp Resource shall be set to the manufacturer default  
2081 value.

## 2082 **8.3 Device Ready-for-OTM State Definition**

2083 The following Resources and their specific properties shall have the value as specified when the  
2084 Device enters ready for ownership transfer:

- 2085 1) The owned Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to  
2086 TRUE.
- 2087 2) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2088 implemented.

- 2089 3) The devowneruid Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2090 4) The deviceid Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is  
2091 implemented. The value of the di Property in /oic/d is undefined.
- 2092 5) The deviceuuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2093 default value.
- 2094 6) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
- 2095 7) The dos of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RFOTM" state  
2096 and dos.p shall equal "FALSE".
- 2097 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

2098 **8.4 Device Ready-for-Provisioning State Definition**

2099 The following Resources and their specific properties shall have the value as specified when the  
2100 Device enters ready for provisioning:

- 2101 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
- 2102 2) The devowneruid Property of the /oic/sec/doxm Resource shall not be nil UUID.
- 2103 3) The deviceuuid Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be set  
2104 to the value that was determined during RFOTM processing. Also the value of the di Property  
2105 in /oic/d Resource shall be the same as the deviceid Property in the /oic/sec/doxm Resource.
- 2106 4) The oxmsel Property of the /oic/sec/doxm Resource shall have the value of the actual OTM  
2107 used during ownership transfer.
- 2108 5) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
- 2109 6) The dos of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFPRO" state and  
2110 dos.p shall equal "FALSE".
- 2111 7) The rowneruid Property of every installed Resource shall be set to a valid Resource owner  
2112 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2113 rowneruid may result in an orphan Resource.
- 2114 8) The /oic/sec/cred Resource shall contain credentials for each entity referenced by an  
2115 rowneruid, amsuuid, devowneruid.

2116 **8.5 Device Ready-for-Normal-Operation State Definition**

2117 The following Resources and their specific properties shall have the value as specified when the  
2118 Device enters ready for normal operation:

- 2119 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
- 2120 2) The devowneruid Property of the /oic/sec/doxm Resource shall not be nil UUID.
- 2121 3) The deviceuuid Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be set  
2122 to the ID that was configured during OTM. Also the value of the "di" Property in /oic/d shall be  
2123 the same as the deviceuuid.
- 2124 4) The oxmsel Property of the /oic/sec/doxm Resource shall have the value of the actual OTM  
2125 used during ownership transfer.
- 2126 5) The isop Property of the /oic/sec/pstat Resource shall be set to TRUE by the Server once  
2127 transition to RFNOP is otherwise complete.
- 2128 6) The dos of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFNOP" state and  
2129 dos.p shall equal "FALSE".
- 2130 7) The rowneruid Property of every installed Resource shall be set to a valid resource owner (i.e.  
2131 an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2132 rowneruid results in an orphan Resource.

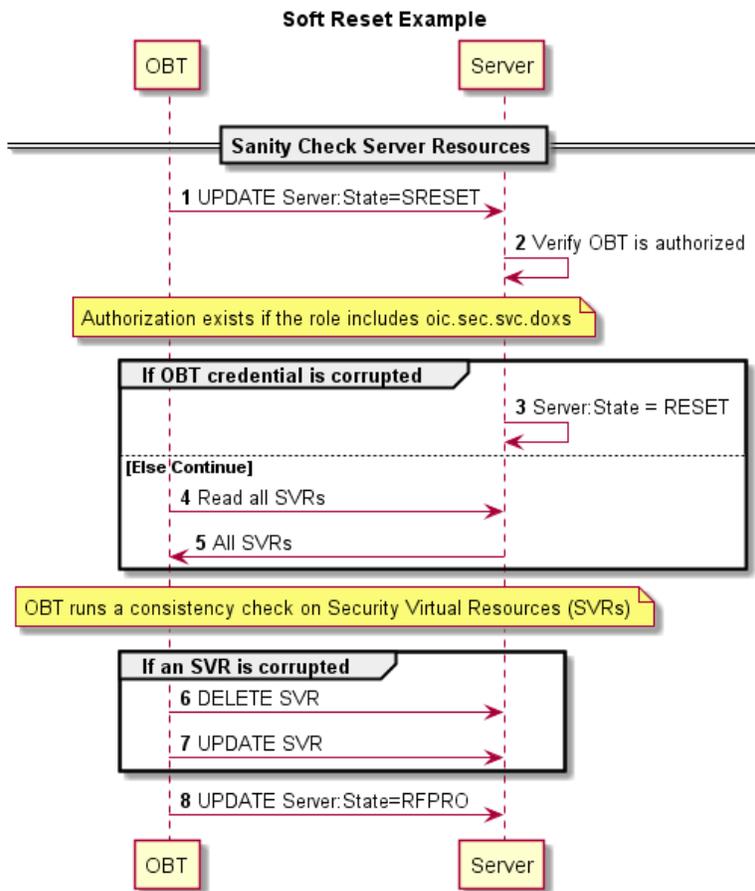
2133 8) The /oic/sec/cred Resource shall contain credentials for each service referenced by a  
 2134 rowneruuid, amsuuid, devowneruuid.

2135 **8.6 Device Soft Reset State Definition**

2136 The soft reset state is defined (e.g. /pstat.dos.s = SRESET) where entrance into this state means  
 2137 the Device is not operational but remains owned by the current owner. The Device may exit  
 2138 SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxxs") using the OC provided during original  
 2139 onboarding (but should not require use of an OTM /doxm.oxxms).

2140 The DOTS should perform a consistency check of the SVR and if necessary, re-provision them  
 2141 sufficiently to allow the Device to transition to RFPRO.

2142 Figure 28 depicts OBT Sanity Check Sequence in SRESET.



2143 **Figure 28 – OBT Sanity Check Sequence in SRESET**

2144

2145 The DOTS should perform a sanity check of SVRs before final transition to RFPRO Device state.  
2146 If the DOTS credential cannot be found or is determined to be corrupted, the Device state  
2147 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to validate  
2148 the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS Devices.

2149 When in SRESET, the following Resources and their specific Properties shall have the values as  
2150 specified.

- 2151 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
- 2152 2) The devowneruuid Property of the /oic/sec/doxm Resource shall remain non-null.
- 2153 3) The devowner Property of the /oic/sec/doxm Resource shall be non-null, if this Property is  
2154 implemented.
- 2155 4) The deviceuuid Property of the /oic/sec/doxm Resource shall remain non-null.
- 2156 5) The deviceid Property of the /oic/sec/doxm Resource shall remain non-null.
- 2157 6) The sct Property of the /oic/sec/doxm Resource shall retain its value.
- 2158 7) The oxmsel Property of the /oic/sec/doxm Resource shall retain its value.
- 2159 8) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
- 2160 9) The /oic/sec/pstat.dos.s Property shall be SRESET.
- 2161 10) The om (operational modes) Property of the /oic/sec/pstat Resource shall be "client-directed  
2162 mode".
- 2163 11) The sm (supported operational modes) Property of /oic/sec/pstat Resource may be updated by  
2164 the Device owner (aka DOTS).
- 2165 12) The rowneruuid Property of /oic/sec/pstat, /oic/sec/doxm, ~~/oic/sec/acl~~, /oic/sec/acl2,  
2166 /oic/sec/amacl, /oic/sec/sacl, and /oic/sec/cred Resources may be reset by the Device owner  
2167 (aka DOTS) and re-provisioned.

2168

## 2169 **9 Security Credential Management**

### 2170 **9.1 Preamble**

2171 This clause provides an overview of the credential types in OCF, along with details of credential  
2172 use, provisioning and ongoing management.

### 2173 **9.2 Credential Lifecycle**

#### 2174 **9.2.1 Credential Lifecycle General**

2175 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)  
2176 issuance and (5) revocation.

#### 2177 **9.2.2 Creation**

2178 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is  
2179 authorized by matching the rowneruuid Property of the /oic/sec/cred resource to the DeviceID of  
2180 the credential the CMS used to establish the secure connection.

2181 Credential Resources created using a CMS may involve specialized credential issuance protocols  
2182 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate  
2183 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of a  
2184 provisioning action by a DOTS, CMS or AMS.

#### 2185 **9.2.3 Deletion**

2186 The CMS should delete known compromised credential Resources. The Device (e.g. the Device  
2187 where the credential Resource is hosted) should delete credential Resources that have expired.

2188 An expired credential Resource may be deleted to manage memory and storage space.

2189 Deletion in OCF key management is equivalent to credential suspension.

#### 2190 **9.2.4 Refresh**

2191 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2192 The /oic/sec/cred Resource supports expiry using the Period Property. Credential refresh may be  
2193 applied when a credential is about to expire or is about to exceed a maximum threshold for bytes  
2194 encrypted.

2195 A credential refresh method specifies the options available when performing key refresh. The  
2196 Period Property informs when the credential should expire. The Device may proactively obtain a  
2197 new credential using a credential refresh method using current unexpired credentials to refresh the  
2198 existing credential. If the Device does not have an internal time source, the current time should be  
2199 obtained from a CMS at regular intervals.

2200 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the CMS  
2201 credentials to the Device. If the onboarding established credentials are allowed to expire the DOTS  
2202 shall re-onboard the Device to re-apply device owner transfer steps.

2203 All Devices shall support at least one credential refresh method.

#### 2204 **9.2.5 Revocation**

2205 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where the  
2206 revocation method involves provisioning of a revocation object that identifies a credential that is to  
2207 be revoked prior to its normal expiration period, a credential Resource is created containing the  
2208 revocation information that supersedes the originally issued credential. The revocation object

2209 expiration should match that of the revoked credential so that the revocation object is cleaned up  
2210 upon expiry.

2211 It is conceptually reasonable to consider revocation applying to a credential or to a Device. Device  
2212 revocation asserts all credentials associated with the revoked Device should be considered for  
2213 revocation. Device revocation is necessary when a Device is lost, stolen or compromised. Deletion  
2214 of credentials on a revoked Device might not be possible or reliable.

### 2215 **9.3 Credential Types**

#### 2216 **9.3.1 Preamble**

2217 The "/oic/sec/cred" Resource maintains a credential type Property that supports several  
2218 cryptographic keys and other information used for authentication and data protection. The  
2219 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric  
2220 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.  
2221 PIN/password).

#### 2222 **9.3.2 Pair-wise Symmetric Key Credentials**

2223 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The  
2224 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2225 Pair-wise keys could be established through ad-hoc key agreement protocols.

2226 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2227 The PublicData Property may contain a token encrypted to the peer Device containing the pair-  
2228 wise key.

2229 The OptionalData Property may contain revocation status.

2230 The Device implementer should apply hardened key storage techniques that ensure the  
2231 PrivateData remains private.

2232 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2233 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2234 unauthorized modifications.

#### 2235 **9.3.3 Group Symmetric Key Credentials**

2236 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are  
2237 used for efficient sharing of data among group participants.

2238 Group keys do not provide authentication of Devices but only establish membership in a group.

2239 The CMS shall provision group symmetric key credentials to the group members. The CMS  
2240 maintains the group memberships.

2241 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2242 The PublicData Property may contain the group name.

2243 The OptionalData Property may contain revocation status.

2244 The Device implementer should apply hardened key storage techniques that ensure the  
2245 PrivateData remains private.

2246 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2247 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2248 unauthorized modifications.

### 2249 **9.3.4 Asymmetric Authentication Key Credentials**

#### 2250 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2251 Asymmetric authentication key credentials contain either a public and private key pair or only a  
2252 public key. The private key is used to sign Device authentication challenges. The public key is used  
2253 to verify a device authentication challenge-response.

2254 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2255 The PublicData Property contains the public key.

2256 The OptionalData Property may contain revocation status.

2257 The Device implementer should apply hardened key storage techniques that ensure the  
2258 PrivateData remains private.

2259 Devices should generate asymmetric authentication key pairs internally to ensure the private key  
2260 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material  
2261 between Devices.

2262 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2263 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2264 unauthorized modifications.

#### 2265 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2266 Devices should employ industry-standard high-assurance techniques when allowing off-device key  
2267 pair creation and provisioning. Use of such key pairs should be minimized, particularly if the key  
2268 pair is immutable and cannot be changed or replaced after provisioning.

2269 When used as part of onboarding, these key pairs can be used to prove the Device possesses the  
2270 manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept  
2271 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the  
2272 Device, and then provisions new OCF Security Domain credentials for use.

### 2273 **9.3.5 Asymmetric Key Encryption Key Credentials**

2274 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when  
2275 distributing or storing the key.

2276 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2277 The PublicData Property contains the public key.

2278 The OptionalData Property may contain revocation status.

2279 The Device implementer should apply hardened key storage techniques that ensure the  
2280 PrivateData remains private.

2281 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2282 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized  
2283 modifications.

2284 **9.3.6 Certificate Credentials**

2285 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a CMS  
2286 or an external certificate authority (CA).

2287 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2288 The issued certificate is stored with the asymmetric key credential Resource.

2289 Other objects useful in managing certificate lifecycle such as certificate revocation status are  
2290 associated with the credential Resource.

2291 Either an asymmetric key credential Resource or a self-signed certificate credential is used to  
2292 terminate a path validation.

2293 The PrivateData Property in the /oic/sec/cred Resource contains the private key.

2294 The PublicData Property contains the issued certificate.

2295 The OptionalData Property may contain revocation status.

2296 The Device implementer should apply hardened key storage techniques that ensure the  
2297 PrivateData remains private.

2298 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2299 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized  
2300 modifications.

2301 **9.3.7 Password Credentials**

2302 Shared secret credentials are used to maintain a PIN or password that authorizes Device access  
2303 to a foreign system or Device that doesn't support any other OCF credential types.

2304 The PrivateData Property in the /oic/sec/cred Resource contains the PIN, password and other  
2305 values useful for changing and verifying the password.

2306 The PublicData Property may contain the user or account name if applicable.

2307 The OptionalData Property may contain revocation status.

2308 The Device implementer should apply hardened key storage techniques that ensure the  
2309 PrivateData remains private.

2310 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2311 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized  
2312 modifications.

2313 **9.4 Certificate Based Key Management**

2314 **9.4.1 Overview**

2315 To achieve authentication and transport security during communications in OCF Security Domain,  
2316 certificates containing public keys of communicating parties and private keys can be used.

2317 The certificate and private key may be issued by a local or remote certificate authority (CA). For  
2318 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of identity.  
2319 In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to validate  
2320 proof of identity and validity.

2321 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,  
2322 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in X.509  
2323 are not supported so that the format intends to meet the constrained Device's requirement.

2324 As for the certificate and CRL management in the Server, the process of storing, retrieving and  
2325 parsing Resources of the certificates and CRL will be performed at the security resource manager  
2326 layer; the relevant interfaces may be exposed to the upper layer.

2327 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of  
2328 certificates and CRL will be stored and managed in SVR database.

2329 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a CMS  
2330 to a Device when it is newly onboarded. The issuing CMS should process certificate revocations  
2331 for certificates it issues. If a certificate private key is compromised, the CMS should revoke the  
2332 certificate. If CRLs are used by a Device, the CMS should regularly (for example; every 3 months)  
2333 update the /oic/sec/crl resource for the Devices it manages.

#### 2334 9.4.2 X.509 Digital Certificate Profiles

##### 2335 9.4.2.1 Digital Certificate Profile General

2336 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in  
2337 IETF RFC 5280.

2338 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications for  
2339 those communities wishing to make use of X.509 technology. The X.509 v3 certificate format is  
2340 described in detail, with additional information regarding the format and semantics of OCF specific  
2341 extension(s). The supported standard certificate extensions are also listed.

2342 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this  
2343 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are  
2344 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,  
2345 compliant entities shall ignore their contents.

2346 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as  
2347 defined in ISO/IEC 8825-1 to encode certificates.

2348 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public  
2349 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs  
2350 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and  
2351 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2352 The following clauses specify the supported standard and custom extensions for the OCF  
2353 certificates profile.

##### 2354 9.4.2.2 Certificate Profile and Fields

###### 2355 9.4.2.2.1 Root CA Certificate Profile

2356 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2357

**Table 16 – X.509 v1 fields for Root CA Certificates**

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by a given CA

Issuer	SHALL match the Subject field
Subject	SHALL match the Issuer field
notBefore	The time at which the Root CA Certificate was generated. See 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation for expiry date. See 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID:1.2.840.10045.3.1.7)

2358 **Table 17** describes X.509 v3 extensions required for Root CA Certificates.

2359 **Table 17 - X.509 v3 extensions for Root CA Certificates**

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	keyCertSign (5) & cRLSign (6) bits shall be enabled. digitalSignature(0) bit may be enabled. All other bits shall not be enabled.
basicConstraints	REQUIRED	Critical	cA = TRUE pathLenConstraint = not present (unlimited)

2360 **9.4.2.2.2 Intermediate CA Certificate Profile**

2361 **Table 18** describes X.509 v1 fields required for Intermediate CA Certificates.

2362 **Table 18 - X.509 v1 fields for Intermediate CA Certificates**

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by Root CA
Issuer	SHALL match the Subject field of the issuing Root CA
Subject	(no stipulation)
notBefore	The time at which the Intermediate CA Certificate was generated. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation for expiry date. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID:1.2.840.10045.3.1.7)

2363 **Table 19** describes X.509 v3 extensions required for Intermediate CA Certificates.

2364

**Table 19 – X.509 v3 extensions for Intermediate CA Certificates**

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	keyCertSign (5) & cRLSign (6) bits shall be enabled. digitalSignature (0) bit may be enabled All other bits shall not be enabled.
basicConstraints	REQUIRED	Critical	cA = TRUE pathLenConstraint = 0 (can only sign End-Entity certs)
certificatePolicies	OPTIONAL	Non-critical	(no stipulation)
cRLDistributionPoints	OPTIONAL	Non-critical	1 or more URIs where the Certificate Revocation List (CRL) from the Root can be obtained.
authorityInformationAccess	OPTIONAL	Non-critical	OCSP URI – the URI of the Root CA's OCSP Responder

2365

**9.4.2.2.3 End-Entity Black Certificate Profile**

2366 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security  
2367 profile.

2368

**Table 20 – X.509 v1 fields for End-Entity Certificates**

V1 Field	Value / Remarks
signatureAlgorithm	ecdsa-with-SHA256 (OID: 1.2.840.10045.4.3.2)
Version	v3 (value is 2)
SerialNumber	SHALL be a positive integer, unique among all certificates issued by the Intermediate CA
Issuer	SHALL match the Subject field of the issuing Intermediate CA
Subject	Subject DN shall include: o=OCF-verified device manufacturer organization name.  The Subject DN may include other attributes (e.g. cn, c, ou, etc.) with no stipulation by OCF.
notBefore	The time at which the End-Entity Certificate was generated. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
notAfter	No stipulation. See clause 10.4.5 for details around IETF RFC 5280-compliant validity field formatting.
Subject Public Key Info	id-ecPublicKey (OID: 1.2.840.10045.2.1) secp256r1 (OID: 1.2.840.10045.3.1.7)

2369 Table 21 describes X.509 v3 extensions required for End-Entity Certificates.

Table 21 – X.509 v3 extensions for End-Entity Certificates

Extension	Required/Optional	Criticality	Value / Remarks
authorityKeyIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdentifier	OPTIONAL	Non-critical	N/A
keyUsage	REQUIRED	Critical	digitalSignature (0) and keyAgreement(4) bits SHALL be the only bits enabled
basicConstraints	OPTIONAL	Non-Critical	cA = FALSE pathLenConstraint = not present
certificatePolicies	OPTIONAL	Non-critical	End-Entity certificates chaining to an OCF Root CA SHOULD contain at least one PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2) corresponding to the version of the OCF Certificate Policy under which it was issued. Additional manufacturer-specific CP OIDs may also be populated.
extendedKeyUsage	REQUIRED	Non-critical	The following extendedKeyUsage (EKU) OIDs SHALL both be present: <ul style="list-style-type: none"> <li>• serverAuthentication - 1.3.6.1.5.5.7.3.1</li> <li>• clientAuthentication - 1.3.6.1.5.5.7.3.2</li> </ul> Exactly ONE of the following OIDs SHALL be present: <ul style="list-style-type: none"> <li>• Identity certificate - 1.3.6.1.4.1.44924.1.6</li> <li>• Role certificate - 1.3.6.1.4.1.44924.1.7</li> </ul> End-Entity certificates SHALL NOT contain the anyExtendedKeyUsage OID (2.5.29.37.0)
subjectAlternativeName	REQUIRED UNDER CERTAIN CONDITIONS	Non-critical	The subjectAltName extension is used to encode one or more Role ID values in role certificates, binding the roles to the subject public key. When the extendedKeyUsage (EKU) extension contains the Identity Certificate OID (1.3.6.1.4.1.44924.1.6), the subjectAltName extension SHOULD NOT be present. If the EKU extension contains the Role Certificate

			OID (1.3.6.1.4.1.44924.1.7), the subjectAltName extension SHALL be present and populated as follows: Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational Unit) components. The OU component, if present, shall specify the authority that defined the semantics of the role. If the OU component is absent, the certificate issuer has defined the role. The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may be present, but shall not be interpreted as roles. The role, and authority shall be encoded as ASN.1 PrintableString type, the restricted character set [0-9a-z-A-z '()+,./:=?].
cRLDistributionPoints	OPTIONAL	Non-critical	1 or more URIs where the Certificate Revocation List (CRL) from the Intermediate CA can be obtained.
authorityInformationAccess	OPTIONAL	Non-critical	OCSP URI – the URI of the Intermediate CA's OCSP Responder
OCF Compliance	OPTIONAL	Non-critical	See 9.4.2.2.4
Manufacturer Usage Description (MUD)	OPTIONAL	Non-critical	Contains a single Uniform Resource Locator (URL) that points to an on-line Manufacturer Usage Description concerning the certificate subject. See 9.4.2.2.5
OCF Security Claims	OPTIONAL	Non-critical	Contains a list of security claims above those required by this OCF Compliance version or Security Profile. See 9.4.2.2.6
OCF CPL Attributes	OPTIONAL	Non-critical	Contains the list of OCF Attributes used to perform OCF Certified Product List lookups

2371 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2372 The OCF Compliance Extension defines required parameters to correctly identify the type of Device,  
2373 its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2374 The extension carries an "ocfVersion" field which provides the specific base version of the OCF  
2375 documents the device implements. The "ocfVersion" field shall contain a sequence of three integers  
2376 ("major", "minor", and "build"). For example, if an entity is certified to be compliant with OCF

2377 specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be set to  
2378 "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote  
2379 compliance to a specified base version of the OCF documents.

2380 The 'securityProfile' field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more  
2381 supported Security Profiles associated with the certificate in string form (UTF-8). All Security  
2382 Profiles associated with the certificate should be identified by this field.

2383 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".  
2384 The fields carry human-readable descriptions of the Device's name and manufacturer, respectively.

2385 The ASN.1 definition of the OCFCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined as  
2386 follows:

```
2387 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2388                               private(4) enterprise(1) OCF(51414) }
2389
2390 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2391
2392 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2393
2394 ocfVersion ::= SEQUENCE {
2395     major    INTEGER,
2396             --Major version number
2397     minor    INTEGER,
2398             --Minor version number
2399     build    INTEGER,
2400             --Build/Micro version number
2401 }
2402
2403 ocfCompliance ::= SEQUENCE {
2404     version          ocfVersion,
2405                     --Device/OCF version
2406     securityProfile  SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2407                     --Sequence of OCF Security Profile OID strings
2408                     --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2409     deviceName       UTF8String,
2410                     --Name of the device
2411     deviceManufacturer UTF8String,
2412                     --Human-Readable Manufacturer
2413                     --of the device
2414 }
```

#### 2415 9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension

2416 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for devices  
2417 to signal to the network the access and network functionality they require to properly function.  
2418 Access controls can be more easily achieved and deployed at scale when the MUD extension is  
2419 used. The current draft of the MUD v3 extension at this time of writing is:

2420 <https://tools.ietf.org/html/draft-ietf-opsawg-mud-15#section-10>

2421 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2422 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2423                          internet(1) security(5) mechanisms(5) pkix(7)
2424                          id-mod(0) id-mod-mudURLExtn2016(88) }
2425
2426 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2427 -- EXPORTS ALL --
2428 IMPORTS
2429     EXTENSION
```

```

2430 FROM PKIX-CommonTypes-2009
2431     { iso(1) identified-organization(3) dod(6) internet(1)
2432       security(5) mechanisms(5) pkix(7) id-mod(0)
2433         id-mod-pkixCommon-02(57) }
2434 id-pe
2435 FROM PKIX1Explicit-2009
2436     { iso(1) identified-organization(3) dod(6) internet(1)
2437       security(5) mechanisms(5) pkix(7) id-mod(0)
2438         id-mod-pkix1-explicit-02(51) } ;
2439 MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2440 ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2441     IDENTIFIED BY id-pe-mud-url }
2442
2443 id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2444
2445 MUDURLSyntax ::= IA5String
2446
2447 END

```

#### 2448 9.4.2.2.6 OCF Security Claims X.509v3 Extension

2449 The OCF Security Claims Extension defines a list of OIDs representing security claims that the  
2450 manufacturer/integrator is making as to the security posture of the device above those required by  
2451 the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2452 The purpose of this extension is to allow for programmatic evaluation of assertions made about  
2453 security to enable some platforms/policies/administrators to better understand what is being  
2454 onboarded or challenged.

2455 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is defined  
2456 as follows:

```

2457 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2458     private(4) enterprise(1) OCF(51414) }
2459
2460 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2461
2462 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2463
2464 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2465 --Device claims that the boot process follows a procedure trusted
2466 --by the firmware and the BIOS
2467
2468 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2469 --Device claims that credentials are stored in a specialized hardware
2470 --protection environment such as a Trusted Platform Module (TPM) or
2471 --similar mechanism.
2472
2473 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2474
2475 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

#### 2476 9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension

2477 The OCF Certified Product List Extension defines required parameters to utilize the OCF  
2478 Compliance Management System Certified Product List (OCMS-CPL). This clause is only  
2479 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to  
2480 verify the compliance level of a device.

2481 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model and  
2482 Version.

2483 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique  
2484 PEN established in the IANA PEN list located at: [https://www.iana.org/assignments/enterprise-](https://www.iana.org/assignments/enterprise-numbers)  
2485 numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as  
2486 reported during OCF Certification.

2487 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field  
2488 found in end-products shall be the same information as reported during OCF Certification.

2489 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found  
2490 in end-products shall be the same information as reported during OCF Certification.

2491 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is defined  
2492 as follows:

```
2493 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2494                               private(4) enterprise(1) OCF(51414) }
2495
2496 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2497
2498     id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2499
2500     cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2501     cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2502     cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2503
2504
2505     ocfCPLAttributes ::= SEQUENCE {
2506         cpl-at-IANAPen      UTF8String,
2507         --Manufacturer's registered IANA Private Enterprise Number
2508         cpl-at-model       UTF8String,
2509         --Device OCF Security Profile
2510         cpl-at-version     UTF8String
2511         --Name of the device
2512     }
```

### 2513 9.4.2.3 Supported Certificate Extensions

2514 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the  
2515 clause number from that RFC to include it by reference. Each extension is summarized here, and  
2516 any modifications to the RFC definition are listed. Devices MUST implement and understand the  
2517 extensions listed here; other extensions from the RFC are not included in this document and  
2518 therefore are not required. 10.4 describes what Devices must implement when validating certificate  
2519 chains, including processing of extensions, and actions to take when certain extensions are absent.

#### 2520 – Authority Key Identifier (4.2.1.1)

2521 The Authority Key Identifier (AKI) extension provides a means of identifying the public key  
2522 corresponding to the private key used to sign a certificate. This document makes the following  
2523 modifications to the referenced definition of this extension:

2524 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier  
2525 sequence are not permitted; only keyIdentifier is allowed. This results in the following grammar  
2526 definition:

```
2527 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2528
2529 AuthorityKeyIdentifier ::= SEQUENCE {
2530     keyIdentifier          [0] KeyIdentifier
2531 }
2532 KeyIdentifier ::= OCTET STRING
```

#### 2533 – Subject Key Identifier (4.2.1.2)

2534 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that  
2535 contain a particular public key.

2536 This document makes the following modification to the referenced definition of this extension:

2537 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's  
2538 SubjectPublicKeyInfo field or a method that generates unique values. This document  
2539 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey  
2540 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains must  
2541 not assume any particular method of computing key identifiers, however, and must only base  
2542 matching AKI's and SKI's in certification path constructions on key identifiers seen in certificates.

2543 – Subject Alternative Name

2544 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role  
2545 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and  
2546 interpreted as described below. When no EKU is present, or has another value, the  
2547 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to  
2548 encode one or more Role ID values in role certificates, binding the roles to the subject public  
2549 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2550 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2551
2552 SubjectAltName ::= GeneralNames
2553
2554 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2555
2556 GeneralName ::= CHOICE {
2557     otherName                [0]     OtherName,
2558     rfc5322Name              [1]     IA5String,
2559     dNSName                  [2]     IA5String,
2560     x400Address              [3]     ORAddress,
2561     directoryName            [4]     Name,
2562     ediPartyName             [5]     EDIPartyName,
2563     uniformResourceIdentifier [6]     IA5String,
2564     iPAddress                [7]     OCTET STRING,
2565     registeredID             [8]     OBJECT IDENTIFIER }
2566
2567     EDIPartyName ::= SEQUENCE {
2568         nameAssigner          [0]     DirectoryString OPTIONAL,
2569         partyName             [1]     DirectoryString }
```

2571 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a  
2572 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name  
2573 shall contain exactly one CN (Common Name) component, and zero or one OU (Organizational  
2574 Unit) components. The OU component, if present, shall specify the authority that defined the  
2575 semantics of the role. If the OU component is absent, the certificate issuer has defined the  
2576 role. The CN component shall encode the role ID. Other GeneralName types in the SEQUENCE may  
2577 be present, but shall not be interpreted as roles. Therefore, if the certificate issuer includes  
2578 non-role names in the subjectAltName extension, the extension should not be marked critical.

2579 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted  
2580 character set [0-9a-z-A-z '()+,./:=?].

2581 – Key Usage (4.2.1.3)

2582 The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing)  
2583 of the key contained in the certificate. The usage restriction might be employed when a key  
2584 that could be used for more than one operation is to be restricted.

2585 This document does not modify the referenced definition of this extension.

2586 – Basic Constraints (4.2.1.9)

2587 The basic constraints extension identifies whether the subject of the certificate is a CA and the  
2588 maximum depth of valid certification paths that include this certificate. Without this extension,  
2589 a certificate cannot be an issuer of other certificates.

2590 This document does not modify the referenced definition of this extension.

2591 – Extended Key Usage (4.2.1.12)

2592  
2593 Extended Key Usage describes allowed purposes for which the certified public key may can be  
2594 used. When a Device receives a certificate, it determines the purpose based on the context of  
2595 the interaction in which the certificate is presented, and verifies the certificate can be used for  
2596 that purpose.

2597 This document makes the following modifications to the referenced definition of this extension:

2598 CAs SHOULD mark this extension as critical.

2599 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).

2600

2601 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2602 – Identity certificate 1.3.6.1.4.1.44924.1.6

2603 – Role certificate 1.3.6.1.4.1.44924.1.7

#### 2604 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2605 See 9.4.3.5 for details.

#### 2606 **9.4.2.5 Encoding of Certificate**

2607 See 9.4.2 for details.

### 2608 **9.4.3 Certificate Revocation List (CRL) Profile**

#### 2609 **9.4.3.1 CRL General**

2610 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use within  
2611 OCF applications for those communities wishing to support revocation features in their PKIs.

2612 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in  
2613 IETF RFC 5280 – Clause 5.1

#### 2614 **9.4.3.2 CRL Profile and Fields**

2615 This clause intentionally left empty.

#### 2616 **9.4.3.3 Encoding of CRL**

2617 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]  
2618 should be used to encode CRL.

#### 2619 **9.4.3.4 CRLs Supported Standard Extensions**

2620 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55] provide  
2621 methods for associating additional attributes with CRLs. The following list of X.509 extensions  
2622 should be supported in this certificate profile:

2623 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides  
2624 a means of identifying the public key corresponding to the private key used to sign a CRL.  
2625 Conforming CRL issuers should use the key identifier method, and shall include this extension  
2626 in all CRLs issued

2627 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that  
2628 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer

2629 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for  
2630 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]  
2631 [X9.55]. Although this document does not provide any recommendation about the use of specific  
2632 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not marked  
2633 critical.

#### 2634 **9.4.3.5 Encryption Ciphers and TLS support**

2635 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support  
2636 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 cipher suite as defined in IETF RFC 7251 and may  
2637 support additional ciphers as defined in the TLS v1.2 specifications.

#### 2638 **9.4.4 Resource Model**

2639 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated  
2640 with a separate crl Resource that is defined for maintaining the revocation list.

2641 The cred Resource contains the certificate information pertaining to the Device. The PublicData  
2642 Property holds the device certificate and CA certificate chain. PrivateData Property holds the  
2643 Device private key paired to the certificate. (See 13.3 for additional detail regarding the  
2644 "/oic/sec/cred" Resource).

2645 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained  
2646 through the CMS. The Device must consider revoked certificates as part of certificate path  
2647 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain a  
2648 full list, the Device must query the CMS for current revocation status. (See 13.4 for additional detail  
2649 regarding the "/oic/sec/crl" Resource).

#### 2650 **9.4.5 Certificate Provisioning**

2651 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have  
2652 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)  
2653 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate shall  
2654 have the format described in 9.4.2.

2655 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,  
2656 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer them  
2657 to the Device including its CA certificate chain. Optionally, the CMS may also transfer one or more  
2658 role certificates, which shall have the format described in Clause 9.4.2. . The subjectPublicKey of  
2659 each role certificate shall match the subjectPublicKey in the Device certificate.

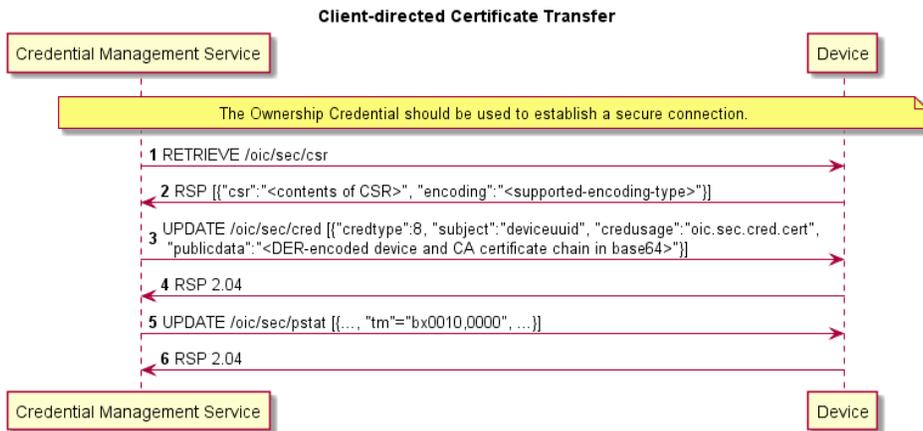
2660 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in  
2661 IETF RFC 2986, and is included here by reference.

2662 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2663 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device  
2664 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.  
2665 The Device determines the public key to present; this may be an already-provisioned key it has  
2666 selected for use with authentication, or if none is present, it may generate a new key pair  
2667 internally and provide the public part. The key pair shall be compatible with the allowed  
2668 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF  
2669 authentication.

2670 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on  
2671 its own, then it is not capable of using certificates. The Device shall advertise this fact both by  
2672 setting the 0x8 bit position in the sct Property of /oic/sec/doxm to 0, and return an error that  
2673 the /oic/sec/csr resource does not exist.

2674 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using the  
 2675 same credid, to maintain the association with the private key. The credential type (oic.sec.cred)  
 2676 used to transfer certificates in Figure 29 is also used to transfer role certificates, by including  
 2677 multiple credentials in the POST from CMS to Device. Identity certificates shall be stored with  
 2678 the credusage Property set to `oic.sec.cred.cert` and role certificates shall be stored with the  
 2679 credusage Property set to `oic.sec.cred.rolecert`.



2680  
 2681 **Figure 29 – Client-directed Certificate Transfer**

2682 **9.4.6 CRL Provisioning**

2683 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the function  
 2684 to register revocation certificates, to sign CRL and to transfer it to Devices.

2685 The CMS sends the CRL to the Device.

2686 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2687 – change of issuer name
- 2688 – change of association between Devices and CA
- 2689 – certificate compromise
- 2690 – suspected compromise of the corresponding private key

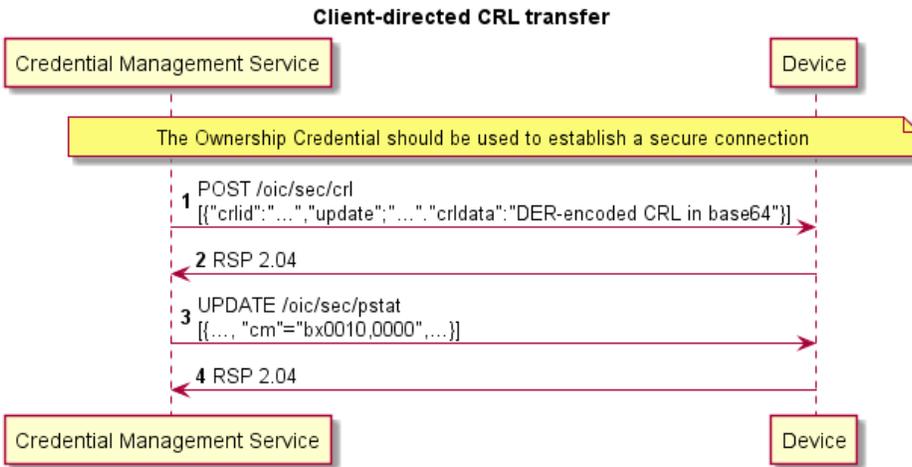
2691 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In some  
 2692 special cases, Devices may request CRL to a given CMS.

2693 There are two options to update and deliver CRL;

- 2694 – CMS pushes CRL to each Device
- 2695 – each Device periodically requests to update CRL

2696 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

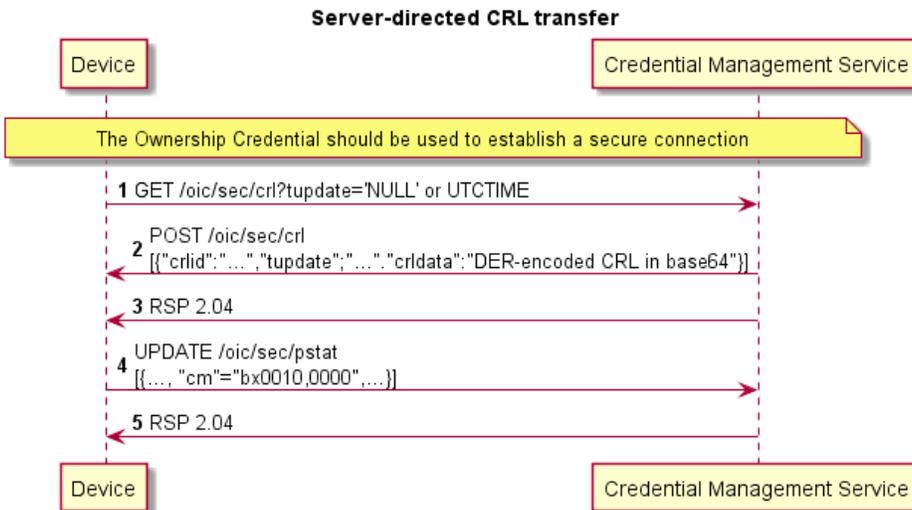
- 2697 1) The CMS may retrieve the CRL Resource Property.
- 2698 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.



2701 **Figure 30 – Client-directed CRL Transfer**

2702 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.

- 2703 1) The Device retrieves the CRL Resource Property update to the CMS.  
 2704 2) If the CMS recognizes the updated CRL information after the designated update time, it may  
 2705 transfer its CRL to the Device.



2706 **Figure 31 – Server-directed CRL Transfer**

2707  
2708

2709 **10 Device Authentication**

2710 **10.1 Device Authentication General**

2711 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the  
2712 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or  
2713 more roles that the server can use in access control decisions. Roles may be asserted when the  
2714 Device authentication is done with certificates.

2715 **10.2 Device Authentication with Symmetric Key Credentials**

2716 When using symmetric keys to authenticate, the Server Device shall include the  
2717 ServerKeyExchange message and set psk\_identity\_hint to the Server's Device ID. The Client shall  
2718 validate that it has a credential with the Subject ID set to the Server's Device ID, and a credential  
2719 type of PSK. If it does not, the Client shall respond with an unknown\_psk\_identity error or other  
2720 suitable error.

2721 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that  
2722 includes a psk\_identity\_hint set to the Client's Device ID. The Server shall verify that it has a  
2723 credential with the matching Subject ID and type. If it does not, the Server shall respond with an  
2724 unknown\_psk\_identity or other suitable error code. If it does, then it shall continue with the DTLS  
2725 protocol, and both Client and Server shall compute the resulting premaster secret.

2726 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2727 When using raw asymmetric keys to authenticate, the Client and the Server shall include a suitable  
2728 public key from a credential that is bound to their Device. Each Device shall verify that the provided  
2729 public key matches the PublicData field of a credential they have, and use the corresponding  
2730 Subject ID of the credential to identify the peer Device.

2731 **10.4 Device Authentication with Certificates**

2732 **10.4.1 Device Authentication with Certificates General**

2733 When using certificates to authenticate, the Client and Server shall each include their certificate  
2734 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.  
2735 Each Device shall validate the certificate chain presented by the peer Device. Each certificate  
2736 signature shall be verified until a public key is found within the /oic/sec/cred Resource with the  
2737 `oic.sec.cred.trustca' credusage. Credential Resource found in /oic/sec/cred are used to terminate  
2738 certificate path validation. Also, the validity period and revocation status should be checked for all  
2739 above certificates, but at this time a failure to obtain a certificate's revocation status (CRL or OCSF  
2740 response) MAY continue to allow the use of the certificate if all other verification checks succeed.

2741 If available, revocation information should be used to verify the revocation status of the certificate.  
2742 The URL referencing the revocation information should be retrieved from the certificate (via the  
2743 authorityInformationAccess or crlDistributionPoints extensions). Other mechanisms may be used  
2744 to gather relevant revocation information like CRLs or OCSF responses.

2745 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2746 Devices must follow the certificate path validation algorithm in Clause 6 of IETF RFC 5280. In  
2747 particular:

- 2748 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is  
2749 present, and that the cA boolean in the extension is TRUE. If either is false, the certificate chain  
2750 MUST be rejected. If the pathLenConstraint field is present, Devices will confirm the number of  
2751 certificates between this certificate and the End-Entity certificate is less than or equal to  
2752 pathLenConstraint. In particular, if pathLenConstraint is zero, only an End-Entity certificate can  
2753 be issued by this certificate. If the pathLenConstraint field is absent, there is no limit to the  
2754 chain length.

- 2755 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is present,  
2756 and that the keyCertSign bit is asserted.
- 2757 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.  
2758 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt  
2759 validation with the public keys of possible issuer certificates whose subject name equals the  
2760 issuer name of this certificate.
- 2761 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage (EKU)  
2762 suitable to the purpose for which it is being presented. An End-Entity certificate which contains  
2763 no ECU extension is not valid for any purpose and must be rejected. Any certificate which  
2764 contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other valid  
2765 EKUs are also present.
- 2766 – Devices MUST verify "transitive ECU" for certificate chains. Issuer certificates (any certificate  
2767 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the certificate  
2768 chain is being presented. An issuer certificate is valid for a purpose if it contains an ECU  
2769 extension and the ECU OID for that purpose is listed in the extension, OR it does not have an  
2770 ECU extension. An issuer certificate SHOULD contain an ECU extension and a complete list of  
2771 EKUs for the purposes for which it is authorized to issue certificates. An issuer certificate  
2772 without an ECU extension is valid for all purposes; this differs from End-Entity certificates  
2773 without an ECU extension.
- 2774 The list of purposes and their associated OIDs are defined in 9.4.2.3.
- 2775 If the Device does not recognize an extension, it must examine the `critical` field. If the field is  
2776 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the  
2777 certificate as if the extension were absent and proceed accordingly. This applies to all certificates  
2778 in a chain.
- 2779 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.
- 2780 **10.4.2 Role Assertion with Certificates**
- 2781 This clause describes role assertion by a client to a server using a certificate role credential. If a  
2782 server does not support the certificate credential type, clients should not attempt to assert roles  
2783 with certificates.
- 2784 Following authentication with a certificate, a client may assert one or more roles by updating the  
2785 server's roles resource with the role certificates it wants to use. The role credentials must be  
2786 certificate credentials and shall include a certificate chain. The server shall validate each certificate  
2787 chain as specified in Clause 10.3. Additionally, the public key in the End-Entity certificate used for  
2788 Device authentication must be identical to the public key in all role (End-Entity) certificates. Also,  
2789 the subject distinguished name in the End-Entity authentication and role certificates must match.  
2790 The roles asserted are encoded in the subjectAltName extension in the certificate. The  
2791 subjectAltName field can have multiple values, allowing a single certificate to encode multiple roles  
2792 that apply to the client. The server shall also check that the ECU extension of the role certificate(s)  
2793 contains the value 1.3.6.1.4.1.44924.1.7 (see Clause 9.4.2.2) indicating the certificate may be used  
2794 to assert roles. Figure 32 describes how a client Device asserts roles to a server.

### Asserting Certificate Role Credentials

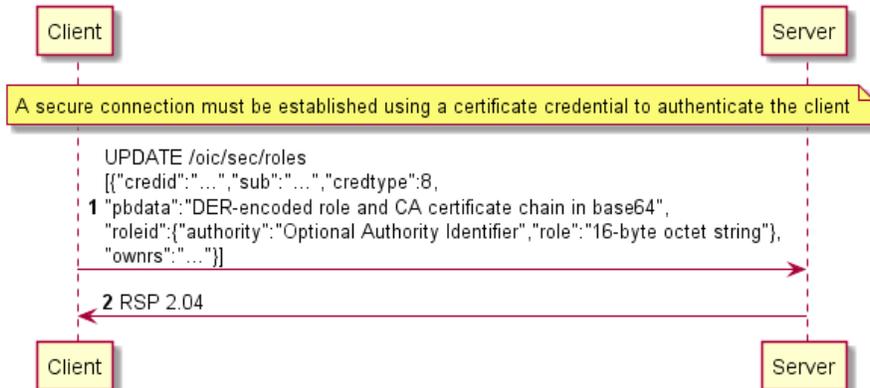


Figure 32 – Asserting a role with a certificate role credential.

2795  
2796

2797 Additional comments for Figure 32

- 2798 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If  
2799 the server does not support certificate credentials, it should return "501 Not Implemented"
- 2800 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration shall  
2801 not exceed the validity period of the role certificate. When fresh CRL information is obtained,  
2802 the certificates in "/oic/sec/roles" should be checked, and the role removed if the certificate is  
2803 revoked or expired.
- 2804 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role  
2805 by a client. It is recommended that servers use the validity period of the certificate as a duration,  
2806 effectively allowing the CMS to decide the duration.
- 2807 4) The format of the data sent in the create call shall be a list of credentials (oic.sec.cred, see  
2808 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not  
2809 be present. For fields that are duplicated in the oic.sec.cred object and the certificate, the  
2810 value in the certificate shall be used for validation. For example, if the Period field is set in the  
2811 credential, the server must treat the validity period in the certificate as authoritative. Similar  
2812 for the roleid data (authority, role).
- 2813 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)
- 2814 6) Clients may GET the /oic/sec/roles resource to determine the roles that have been previously  
2815 asserted. An array of credential objects shall be returned. If there are no valid certificates  
2816 corresponding to the currently connected and authenticated Client's identity, then an empty  
2817 array (i.e. []) shall be returned.

#### 2818 10.4.3 OCF PKI Roots

2819 This clause intentionally left empty.

#### 2820 10.4.4 PKI Trust Store

2821 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store the  
2822 OCF Root CA certificates in the oic/sec/cred resource and SHOULD physically store this resource  
2823 in a hardened memory location where the certificates cannot be tampered with.

2824 **10.4.5 Path Validation and extension processing**

2825 Devices SHALL follow the certificate path validation algorithm in Clause 6 of IETF RFC 5280. In  
2826 addition, the following are best practices and SHALL be adhered to by any OCF-compliant  
2827 application handling digital certificates

2828 – Validity Period checking

2829 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and  
2830 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,  
2831 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2832 – Revocation checking

2833 Relying applications SHOULD check the revocation status for all certificates, but at this time,  
2834 an application MAY continue to allow the use of the certificate upon a failure to obtain a  
2835 certificate's revocation status (CRL or OCSP response), if all other verification checks succeed.

2836 – basicConstraints

2837 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify that  
2838 the basicConstraints extension is present, flagged critical, and that the cA boolean value in the  
2839 extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2840 If the pathLenConstraint field is present, Devices will confirm the number of certificates between  
2841 this certificate and the End-Entity certificate is less than or equal to pathLenConstraint. In  
2842 particular, if pathLenConstraint is zero, only an End-Entity certificate can be issued by this  
2843 certificate. If the pathLenConstraint field is absent, there is no limit to the chain length.

2844 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged  
2845 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a  
2846 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a  
2847 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a  
2848 null value.

2849 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-compliant  
2850 Root CA certificates SHALL NOT contain a pathLenConstraint. This allows additional tiers of  
2851 Intermediate CAs to be implemented in the future without changing the Root CA trust anchors,  
2852 should such a requirement emerge.

2853 – keyUsage

2854 For all certificates, Devices shall verify that the key usage extension is present and flagged  
2855 critical.

2856 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits SHALL  
2857 be asserted.

2858 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be  
2859 asserted.

2860 – extendedKeyUsage:

2861 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be  
2862 rejected.

2863 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)  
2864 are required for compatibility with various TLS implementations.

2865 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)  
2866 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be  
2867 present and End-Entity certificates with EKU extensions containing both OIDs SHALL be  
2868 rejected.

2869 – certificatePolicies

2870 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one  
2871 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)  
2872 corresponding to the version of the OCF Certificate Policy under which it was issued. Additional  
2873 manufacturer-specific CP OIDs may also be populated.

## 2874 **10.5 Device Authentication with OCF Cloud**

### 2875 **10.5.1 Device Authentication with OCF Cloud General**

2876 The mechanisms for Device Authentication in clauses 10.2, 10.3 and 10.4 imply that a Device is  
2877 authorized to communicate with any other Device meeting the criteria provisioned in /oic/sec/cred;  
2878 the "/oic/sec/acl2" Resource ~~(or "/oic/sec/acl1" resource of OIC1.1 Servers) are is~~ additionally used  
2879 to restrict access to specific Resources. The present clause describes Device authentication for  
2880 OCF Cloud, which uses slightly different criteria as described in clause 5. A Device accessing an  
2881 OCF Cloud shall establish a TLS session. The mutual authenticated TLS session is established  
2882 using Server certificate and Client certificate.

2883 Each Device is identified based on the Access Token it is assigned during Device Registration.  
2884 The OCF Cloud holds an OCF Cloud association table that maps Access Token, User ID and Device  
2885 ID. The Device Registration shall happen while the Device is in RFNOP state. After Device  
2886 Registration, the updated Access Token, Device ID and User ID are used by the Device for the  
2887 subsequent connection with the OCF Cloud.

### 2888 **10.5.2 Device Connection with the OCF Cloud**

2889 The Device should establish the TLS connection using the certificate based credential. The  
2890 connection should be established after Device is provisioned by Mediator.

2891 The TLS session is established between Device and the OCF Cloud as specified in IETF RFC 8323.  
2892 The OCF Cloud is expected to provide certificate signed by trust anchor that is present in cred  
2893 entries of the Device. These cred entries are expected to be configured by the Mediator.

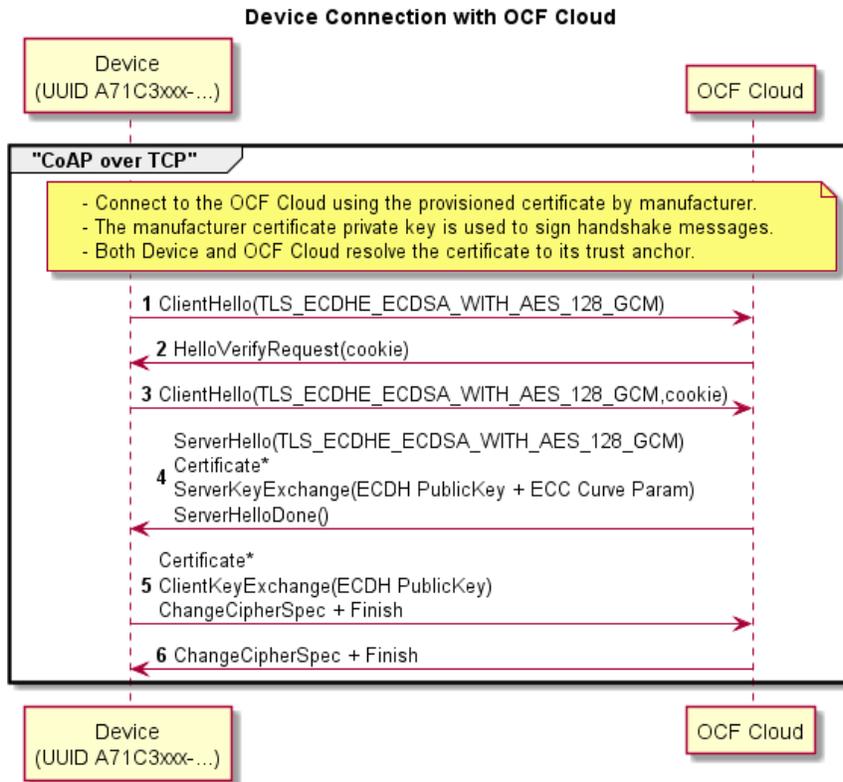
2894 The Device shall validate the OCF Cloud's identity based on the credentials that are contained in  
2895 /oic/sec/cred Resource entries of the Device.

2896 The OCF Cloud is expected to validate the manufacturer certificate provided by the Device.

2897 The assumption is that the OCF Cloud User trusts the OCF Cloud that the Device connects. The  
2898 OCF Cloud connection should not happen without the consent of the OCF Cloud User. The  
2899 assumption is that the OCF Cloud User has either service agreement with the OCF Cloud provider  
2900 or uses manufacturer provided OCF Cloud.

2901 If authentication fails, the "clec" Property of oic.r.coapcloudconf Resource on the Device shall be  
2902 updated about the failed state, if it is supported by the Device. If authentication succeeds, the  
2903 Device and OCF Cloud should establish an encrypted link in accordance with the negotiated cipher  
2904 suite.

2905 Figure 33 depicts sequence for Device connection with OCF Cloud and steps described in Table 22.



**Table 22 – Device connection with the OCF Cloud flow**

Steps	Description
1 - 6	TLS connection between the OCF Cloud and Device. The Device's manufacturer certificate may contain data attesting to the Device hardening and security properties

**10.5.3 Security Considerations**

When an OCF Server receives a request sent via the OCF Cloud, then the OCF Server permits that request using the identity of the OCF Cloud rather than the identity of the OCF Client. If there is no mechanism through which the OCF Cloud permits only those interactions which the user intends between OCF Clients and OCF Server via the OCF Cloud, and denies all other interactions, then OCF Clients might get elevated privileges by submitting a request via the OCF Cloud. This is highly undesirable from the security perspective. Consequently, OCF Cloud implementations are expected to provide some mechanism through which the OCF Cloud prevents OCF Clients getting elevated privileges when submitting a request via the OCF Cloud. In the present document release, the details of the mechanism are left to the implementation.

2919 The security considerations about the manufacturer certificate as described in 7.3.6.5 are also  
2920 applicable in the Device authentication with the OCF Cloud.

2921 The Device should validate the OCF Cloud's TLS certificate as defined by IETF RFC 6125 and in  
2922 accordance with its requirements for Server identity authentication.

2923 The "uid" and "di" Property Value of "/oic/d" Resource may be considered personally identifiable  
2924 information in some regulatory regions, and the OCF Cloud is expected to provide protections  
2925 appropriate to its governing regulatory bodies.

2926

2927 **11 Message Integrity and Confidentiality**

2928 **11.1 Preamble**

2929 Secured communications between Clients and Servers are protected against eavesdropping,  
2930 tampering, or message replay, using security mechanisms that provide message confidentiality and  
2931 integrity.

2932 **11.2 Session Protection with DTLS**

2933 **11.2.1 DTLS Protection General**

2934 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices  
2935 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See  
2936 11.3 for a list of required and optional cipher suites for message communication.

2937 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1  
2938 or lower.

2939 Multicast session semantics are not yet defined in this version of the security document.

2940 **11.2.2 Unicast Session Semantics**

2941 For unicast messages between a Client and a Server, both Devices shall authenticate each other.  
2942 See Clause 10 for details on Device Authentication.

2943 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.  
2944 The sending Device shall encrypt and authenticate messages as defined by the selected cipher  
2945 suite and the receiving Device shall verify and decrypt the messages before processing them.

2946 **11.2.3 Cloud Session Semantics**

2947 The messages between the OCF Cloud and Device shall be exchanged only if the Device and OCF  
2948 Cloud authenticate each other as described in 10.4.3. The asymmetric cipher suites as described  
2949 in 11.3.5 shall be employed for establishing a secured session and for encrypting/decrypting  
2950 between the OCF Cloud and the Device. The OCF Endpoint sending the message shall encrypt  
2951 and authenticate the message using the cipher suite as described in 11.3.5 and the OCF Endpoint  
2952 shall verify and decrypt the message before processing it.

2953 **11.3 Cipher Suites**

2954 **11.3.1 Cipher Suites General**

2955 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher  
2956 suites allowed during ownership transfer and normal operation. The following RFCs provide  
2957 additional information about the cipher suites used in OCF.

2958 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2959 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2960 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and  
2961 PSKs

2962 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2963 **11.3.2 Cipher Suites for Device Ownership Transfer**

2964 **11.3.2.1 Just Works Method Cipher Suites**

2965 The Just Works OTM may use the following (D)TLS cipher suites.

2966 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,

2967 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

2968 All Devices supporting Just Works OTM shall implement:

2969 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256 (with the value 0xFF00)

2970 All Devices supporting Just Works OTM should implement:

2971 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256 (with the value 0xFF01)

2972 **11.3.2.2 Random PIN Method Cipher Suites**

2973 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2974 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2975 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2976 All Devices supporting Random Pin Based OTM shall implement:

2977 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256

2978 **11.3.2.3 Certificate Method Cipher Suites**

2979 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2980 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

2981 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2982 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2983 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2984 Using the following curve:

2985 secp256r1 (See IETF RFC 4492)

2986 All Devices supporting Manufacturer Certificate Based OTM shall implement:

2987 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2988 Devices supporting Manufacturer Certificate Based OTM should implement:

2989 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2990 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2991 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2992 **11.3.3 Cipher Suites for Symmetric Keys**

2993 The following cipher suites are defined for (D)TLS communication using PSKs:

2994 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2995 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2996 TLS\_PSK\_WITH\_AES\_128\_CCM\_8, (\* 8 OCTET Authentication tag \*)

2997 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

2998 TLS\_PSK\_WITH\_AES\_128\_CCM, (\* 16 OCTET Authentication tag \*)

2999 TLS\_PSK\_WITH\_AES\_256\_CCM,

3000 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

3001 All Devices shall implement the following:

3002 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

3003

3004 Devices should implement the following:

3005 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

3006 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

3007 TLS\_PSK\_WITH\_AES\_128\_CCM\_8,

3008 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

3009 TLS\_PSK\_WITH\_AES\_128\_CCM,

3010 TLS\_PSK\_WITH\_AES\_256\_CCM

#### 3011 **11.3.4 Cipher Suites for Asymmetric Credentials**

3012 The following cipher suites are defined for (D)TLS communication with asymmetric keys or  
3013 certificates:

3014 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

3015 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

3016 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

3017 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

3018 Using the following curve:

3019 secp256r1 (See IETF RFC 4492)

3020 All Devices supporting Asymmetric Credentials shall implement:

3021 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

3022 All Devices supporting Asymmetric Credentials should implement:

3023 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

3024 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

3025 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

#### 3026 **11.3.5 Cipher suites for OCF Cloud Credentials**

3027 The following cipher suites are defined for TLS communication with certificates:

3028 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,

3029 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,

3030 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,

3031 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,

3032 TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,

3033 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

3034 All Devices supporting OCF Cloud Certificate Credentials shall implement:

3035 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256

3036 All Devices supporting OCF Cloud Certificate Credentials should implement:

3037 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,

3038 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,

3039 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,  
3040 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,  
3041 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  
3042

3043 **12 Access Control**

3044 **12.1 ACL Generation and Management**

3045 This clause will be expanded in a future version of the document.

3046 **12.2 ACL Evaluation and Enforcement**

3047 **12.2.1 ACL Evaluation and Enforcement General**

3048 The Server enforces access control over application Resources before exposing them to the  
3049 requestor. The Security Layer in the Server authenticates the requestor when access is received  
3050 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL  
3051 entries that specify the requestor's identity, role or may match authenticated requestors using a  
3052 subject wildcard.

3053 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a  
3054 subject wildcard match of anonymous requestors.

3055 Access is denied if a requested Resource is not matched by an ACL entry.

3056 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources  
3057 may be granted prior to provisioning of ACL Resources.

3058 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries (ACE2)  
3059 that employ a Resource matching algorithm that uses an array of Resource references to match  
3060 Resources to which the ACE2 access policy applies. Matching consists of comparing the values of  
3061 the ACE2 "resources" Property (see Clause 13) to the requested Resource. Resources are matched  
3062 in two ways:

- 3063 1) host reference (href)
- 3064 2) resource wildcard (wc).

3065 **12.2.2 Host Reference Matching**

3066 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for  
3067 Resource matching.

- 3068 – The href Property shall be used to find an exact match of the Resource name if present.

3069 **12.2.3 Resource Wildcard Matching**

3070 When present, a wildcard (wc) expression shall be used to match multiple Resources using a  
3071 wildcard Property contained in the oic.sec.ace2.resource-ref structure.

3072 A wildcard expression may be used to match multiple Resources using a wildcard Property  
3073 contained in the oic.sec.ace2.resource-ref structure. The wildcard matching strings are defined in  
3074 Table 23.

3075 **Table 23 – ACE2 Wildcard Matching Strings Description**

String	Description
"+"	Shall match all Discoverable Non-Configuration Resources which expose at least one Secure OCF Endpoint.
"-"	Shall match allDiscoverable Non-Configuration Resources which expose at least one Unsecure OCF Endpoint.
"**"	Shall match all Non-Configuration Resources.

3076 NOTE Discoverable resources appear in the "/oic/wk/res" Resource, while non-discoverable resources may appear in  
3077 other collection resources but do not appear in the /res collection.

3078 **12.2.4 Multiple Criteria Matching**

3079 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for  
3080 each array element. For example, if a first array element of the "resources" Property contains  
3081 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",  
3082 then Resources that match either of the two 'href' criteria shall be included in the set of matched  
3083 Resources.

3084 **Example 1 JSON for Resource matching**

```
3085 {  
3086 //Matches Resources named "/x/door1" or "/x/door2"  
3087 "resources": [  
3088   {  
3089     "href": "/x/door1"  
3090   },  
3091   {  
3092     "href": "/x/door2"  
3093   },  
3094 ]  
3095 }
```

3096 **Example 2 JSON for Resource matching**

```
3097 {  
3098 // Matches all Resources  
3099 "resources": [  
3100   {  
3101     "wc": ""  
3102   }  
3103 ]  
3104 }
```

3105 **12.2.5 Subject Matching using Wildcards**

3106 When the ACE subject is specified as the wildcard string "" any requestor is matched. The OCF  
3107 server may authenticate the OCF client, but is not required to.

3108 **Examples: JSON for subject wildcard matching**

```
3109 //matches all subjects that have authenticated and confidentiality protections in place.  
3110 "subject" : {  
3111   "conntype" : "auth-crypt"  
3112 }  
3113 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
3114 "subject" : {  
3115   "conntype" : "anon-clear"  
3116 }
```

3117 **12.2.6 Subject Matching using Roles**

3118 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 3119 1) The requestor authenticated with a symmetric key credential, and the role is present in the  
3120 roleid Property of the credential's entry in the credential resource, or

3121 2) The requestor authenticated with a certificate, and a valid role certificate is present in the roles  
3122 resource with the requestor's certificate's public key at the time of evaluation. Validating role  
3123 certificates is defined in 10.3.1.

## 3124 **12.2.7 ACL Evaluation**

### 3125 **12.2.7.1 ACE2 matching algorithm**

3126 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 3127 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these ACE2  
3128 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the signature,  
3129 at least once, following update of the "/oic/sec/sacl" Resource.
- 3130 2) The local /oic/sec/acl2 Resource contributes its ACE2 entries for matching.
- 3131 3) Access shall be granted when all these criteria are met:
  - 3132 a) The requestor is matched by the ACE2 "subject" Property.
  - 3133 b) The requested Resource is matched by the ACE2 resources PropertyProperty and the  
3134 requested Resource shall exist on the local Server.
  - 3135 c) The "period" Property constraint shall be satisfied.
  - 3136 d) The "permission" Property constraint shall be applied.

3137 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching  
3138 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=---UDN; Then UNION  
3139 (Perm1, Perm2)=CRUDN.

3140 The Server shall enforce access based on the effective permissions granted.

3141 Batch requests to Resource containing Links require additional considerations when accessing the  
3142 linked Resources. ACL considerations for batch request to the Atomic Measurement Resource  
3143 Type are provided in clause 12.2.7.2. ACL considerations for batch request to the Collection  
3144 Resource Type are provided in 12.2.7.3.

### 3145 **12.2.7.2 ACL considerations for batch request to the Atomic Measurement Resource Type**

3147 The present clause shall apply to any Resource Type based on the Atomic Measurement Resource  
3148 Type.

3149 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only  
3150 local references and there is an ACE matching the Atomic Measurement Resource which permits  
3151 the request, then the corresponding requests to the linked Resources of the Atomic Measurement  
3152 Resource shall be permitted by the OCF Server. That is, the request to each linked Resource is  
3153 permitted regardless of whether there is an ACE configured on the OCF Server which would permit  
3154 a corresponding request from the OCF Client (which sent the batch request to the Atomic  
3155 Measurement Resource) addressing the linked Resource.

### 3156 **12.2.7.3 ACL considerations for batch request to the Collection Resource Type**

3157 The present clause shall apply to any Resource Type based on the Collection Resource Type.

3158 If an OCF Server receives a batch request to a Collection Resource containing only local references  
3159 and there is an ACE matching the Collection Resource which permits the request, then the  
3160 corresponding requests to the linked Resources of the Collection Resource shall be permitted by  
3161 the OCF Server. That is, the request to each linked Resource is permitted regardless of whether  
3162 there is an ACE configured on the OCF Server which would permit a corresponding request from  
3163 the OCF Client (which sent the batch request to the Collection Resource) addressing the linked  
3164 Resource.



3166 **13 Security Resources**

3167 **13.1 Security Resources General**

3168 OCF Security Resources are shown in Figure 34.

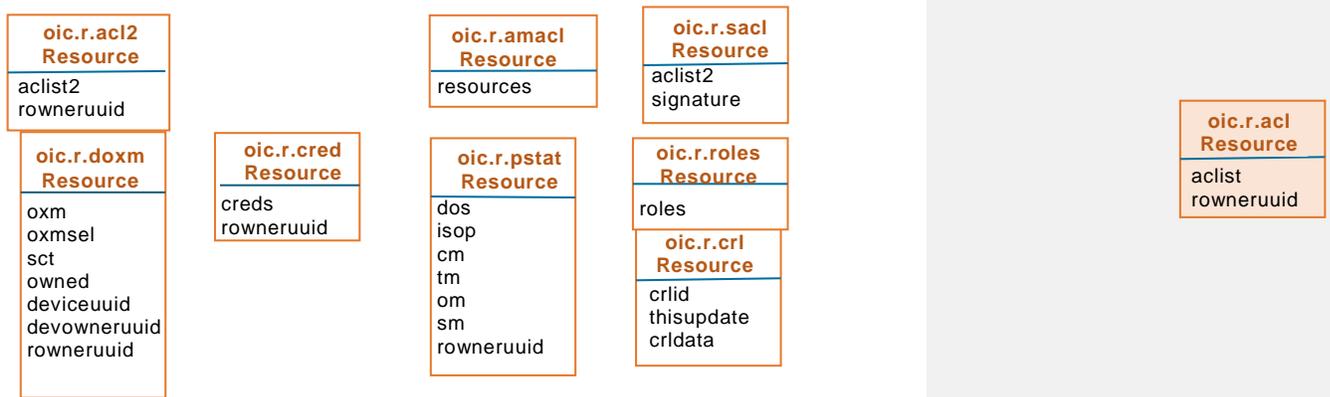
3169 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

3170 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

3171 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

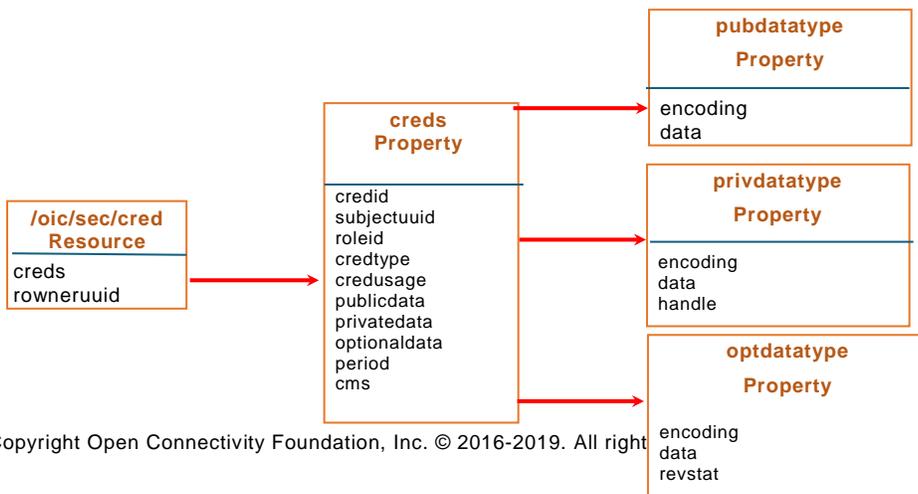
3172 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

3173



3174

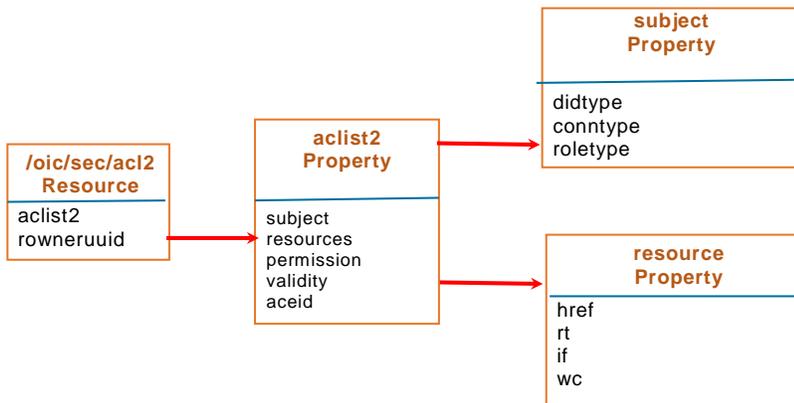
**Figure 34 – OCF Security Resources**



3175

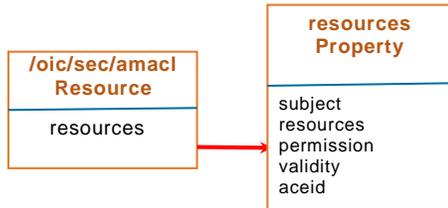
Figure 35 – /oic/sec/cred Resource and Properties

3176



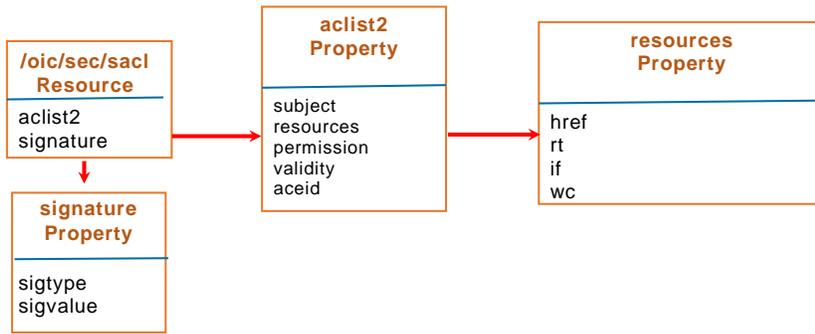
3177

Figure 36 – /oic/sec/acl2 Resource and Properties



3178

Figure 37 – /oic/sec/amacl Resource and Properties



3179

Figure 38 – /oic/sec/sacl Resource and Properties

3180 **13.2 Device Owner Transfer Resource**

3181 **13.2.1 Device Owner Transfer Resource General**

3182 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

3183 Resource discovery processing respects the CRUDN constraints supplied as part of the security  
3184 Resource definitions contained in this document.

3185 "/oic/sec/doxm" Resource is defined in Table 24.

3186

**Table 24 – Definition of the /oic/sec/doxm Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/doxm	Device OTMs	oic.r.doxm	oic.if.baseline	Resource for supporting Device owner transfer	Configuration

3187 Table 25 defines the Properties of the "/oic/sec/doxm" Resource.

3188

**Table 25 – Properties of the /oic/sec/doxm Resource**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
OTM	oxms	oic.sec.doxmtype	array	Yes		R	Value identifying the owner-transfer-method and the organization that defined the method.
OTM Selection	oxmsel	oic.sec.doxmtype	UINT16	Yes	RESET	R	Server shall set to (4) "oic.sec.oxm.self"
					RFOTM	RW	DOTS shall set to it's selected DOTS and both parties execute the DOTS. After secure owner transfer session is established DOTS shall update the oxmsel again making it permanent. If the DOTSDOXS fails the Server shall transition device state to RESET.
					RFPRO	R	n/a
					RFNOP	R	n/a
Supported Credential Types	sct	oic.sec.credtype	bitmask	Yes		R	Identifies the types of credentials the Device supports. The Server sets this value at framework initialization after determining security capabilities.
					RESET	R	Server shall set to FALSE.
					RFOTM	RW	DOTS shall set to TRUE after secure owner transfer session is established..
					RFPRO	R	n/a
Device Ownership Status	owned	Boolean	T F	Yes		R	Server shall set to FALSE.
					RFOTM	RW	DOTS shall set to TRUE after secure owner transfer session is established..
					RFPRO	R	n/a
					RFNOP	R	n/a
Device UUID	deviceuuid	String	oic.sec.didtype	Yes		R	Server shall construct a temporary random UUID that differs for each transition to RESET.
					RFOTM	RW	DOTS shall update to a value it has selected after secure owner transfer session is established. If update fails with error PROPERTY_NOT_FOUND the DOTS shall either accept the Server provided value or update /doxm.owned=FALSE and terminate the session.
					RFPRO	R	n/a
					RFNOP	R	n/a
						R	n/a

Device Owner Id	devowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
					RFOTM	RW	DOTS shall set value after secure owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	R	n/a
Resource Owner Id	rowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
					RFOTM	RW	The DOTS shall configure the rowneruuid Property when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOTS (referenced via devowneruuid Property) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid does not refer to a valid DOTS device identifier the Server shall transition to RESET Device state.

3189 **Table 26** defines the Properties of the "/oic/sec/didtype".

3190 **Table 26 – Properties of the /oic/sec/didtype Property**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Device ID	uuid	String	uuid	Yes	RW	-	A uuid value

3191 The oxms Property contains a list of OTM where the entries appear in the order of preference. This  
3192 Property contains the higher priority methods appearing before the lower priority methods. The  
3193 DOTS queries this list at the time of onboarding and selects the most appropriate method.

3194 The DOTS shall update the oxmsel Property of the /oic/sec/doxm Resource with the OTM that was  
3195 used to onboard the Device.

3196 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3197 <DoxmType> ::= <NSS>
3198 <NSS> ::= <Identifier> | { {<NID> "." } <NameSpaceQualifier> "." } <Method>
3199 <NID> ::= <Vendor-or-Organization>
3200 <Identifier> ::= INTEGER
3201 <NameSpaceQualifier> ::= String
3202 <Method> ::= String
3203 <Vendor-Organization> ::= String

```

3204 When an OTM successfully completes, the *owned* Property is set to '1' (TRUE). Consequently,  
3205 subsequent attempts to take ownership of the Device will fail.

3206 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in the  
3207 "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource is  
3208 UPDATED to non-nil UUID value.

3209 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource  
3210 after it has updated the devowneruuid Property value of the /oic/sec/doxm Resource to a non-nil-  
3211 UUID value.

3212 The Device vendor shall determine that the Device identifier (deviceuuid) is persistent (not  
3213 updatable) or that it is non-persistent (updatable by the owner transfer service – a.k.a DOTS).

3214 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall  
3215 fail with the error PROPERTY\_NOT\_FOUND.

3216 If the deviceuuid Property of the "/oic/sec/doxm" Resource is non-persistent, the request to  
3217 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is  
3218 RESET. If the UPDATE to deviceuuid Property of the /oic/sec/doxm Resource fails while in the  
3219 RFOTM Device state the device state shall transition to RESET where the Server shall set the  
3220 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g. "00000000-  
3221 0000-0000-0000-000000000000").

3222 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the  
3223 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the deviceuuid  
3224 Property of the "/oic/sec/doxm" Resource each time the device enters RESET Device state. The  
3225 temporary deviceuuid value is used while the device state is in the RESET state and while in the  
3226 RFOTM device state until the DOTS establishes a secure OTM connection. The DOTS should  
3227 RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm" Resource after it has  
3228 updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-nil-UUID value.

3229 The deviceuuid Property of the "/oic/sec/doxm" Resource shall expose a persistent value(i.e. is  
3230 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTSDOXS  
3231 via an OCF Interface to the deviceuuid Property of the /oic/sec/doxm Resource during RFOTM  
3232 Device state.).

3233 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a  
3234 secure OTM connection and UPDATES the devowneruuid Property to a non-nil UUID value.  
3235 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the deviceuuid Property of  
3236 the /oic/sec/doxm Resource shall reveal the persistent or semi-persistent value to authenticated  
3237 requestors and shall reveal the temporary non-repeated value to unauthenticated requestors.

3238 See 13.16 for additional details related to privacy sensitive considerations.

### 3239 **13.2.2 Persistent and Semi-persistent Device Identifiers**

3240 The Device vendor determines whether a device identifier can be set by a configuration tool or  
3241 whether it is immutable. If it is an immutable value this document refers to it as a persistent device  
3242 identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four device  
3243 identifiers that could be considered persistent or semi-persistent :

- 3244 1) "deviceuuid" Property of "/oic/sec/doxm"
- 3245 2) "di" Property of "/oic/d"
- 3246 3) "piid" Property of "/oic/d"
- 3247 4) "pi" Property of "/oic/p"

### 3248 **13.2.3 Onboarding Considerations for Device Identifier**

3249 The deviceuuid is used to onboard the Device. The other identifiers (di, piid and pi) are not essential  
3250 for onboarding. The onboarding service (aka DOTS) may not know a priori whether the Device to  
3251 be onboarded is using persistent or semi-persistent identifiers. An OCF Security Domain owner  
3252 may have a preference for persistent or semi-persistent device identifiers. Detecting whether the  
3253 Device is using persistent or semi-persistent deviceuuid can be achieved by attempting to update  
3254 it.

3255 If the "deviceuuid" Property of the /oic/sec/doxm Resource is persistent, then an UPDATE request,  
 3256 at the appropriate time during onboarding shall fail with an appropriate error response.

3257 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device  
 3258 state is RFOTM and when devowneruuid Property value of the /oic/sec/doxm Resource has a non-  
 3259 nil UUID value.

3260 If the "deviceuuid" Property of the /oic/sec/doxm Resource is semi-persistent, subsequent to a  
 3261 successful UPDATE request to change it; the Device shall remember the semi-persistent value  
 3262 until the next successful UPDATE request or until the Device state transitions to RESET.

3263 See 13.16 for addition behavior regarding "deviceuuid".

3264

3265 **13.2.4 OCF defined OTMs**

3266 Table 27 defines the Properties of the "oic.sec.doxmtype".

3267

**Table 27 – Properties of the oic.sec.doxmtype Property**

Value Type Name	Value Type URN (optional)	Enumeration Value (mandatory)	Description
OCFJustWorks	oic.sec.doxm.jw	0	The just-works method relies on anonymous Diffie-Hellman key agreement protocol to allow an DOTS to assert ownership of the new Device. The first DOTS to make the assertion is accepted as the Device owner. The just-works method results in a shared secret that is used to authenticate the Device to the DOTS and likewise authenticates the DOTS to the Device. The Device allows the DOTS to take ownership of the Device, after which a second attempt to take ownership by a different DOTS will fail <sup>a</sup> .
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a DOTSDOXS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTSDOXS signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec.doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.doxm.self	4	The manufacturer shall set the /doxm.oxmsel value to (4). The Server shall reset this value to (4) upon entering RESET Device state.
OCF Reserved	<Reserved>	5-0xFEFF	Reserved for OCF use
Vendor-defined Value Type Name	<Reserved>	0xFF00-0xFFFF	Reserved for vendor-specific OTM use

<sup>a</sup> The just-works method is subject to a man-in-the-middle attacker. Precautions should be taken to provide physical security when this method is used.

3268 **13.3 Credential Resource**

3269 **13.3.1 Credential Resource General**

3270 The /oic/sec/cred Resource maintains credentials used to authenticate the Server to Clients and  
3271 support services as well as credentials used to verify Clients and support services.

3272 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared  
3273 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to  
3274 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3275 In order to provide an interface which allows management of the "creds" Array Property, the  
3276 RETRIEVE, UPDATE and DELETE operations on the oic.r.cred Resource shall behave as follows:

- 3277 1) A RETRIEVE shall return the full Resource representation, except that any write-only Properties  
3278 shall be omitted (e.g. private key data).
- 3279 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3280 UPDATE request, as follows:
- 3281 a) If an UPDATE representation includes the "creds" array Property, then:
- 3282 i) Supplied creds with a "credid" that matches an existing "credid" shall replace completely  
3283 the corresponding cred in the existing "creds" array.
- 3284 ii) Supplied creds without a "credid" shall be appended to the existing "creds" array, and  
3285 a unique (to the cred Resource) "credid" shall be created and assigned to the new cred  
3286 by the Server. The "credid" of a deleted cred should not be reused, to improve the  
3287 determinism of the interface and reduce opportunity for race conditions.
- 3288 iii) Supplied creds with a "credid" that does not match an existing "credid" shall be  
3289 appended to the existing "creds" array, using the supplied "credid".
- 3290 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device  
3291 States in which an UPDATE of the "creds" array Property is always rejected. If OCF  
3292 Device is in a Device State where the Access Mode in this row contains "R", then the  
3293 OCF Device shall reject all UPDATES of the "creds" array Property.
- 3294 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not remove  
3295 the oic.r.cred Resource.
- 3296 4) A DELETE with one or more "credid" query parameters shall remove the cred(s) with the  
3297 corresponding credid(s) from the "creds" array.
- 3298 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in  
3299 which a DELETE is always rejected. If OCF Device is in a Device State where the Access Mode  
3300 in this row contains "R", then the OCF Device shall reject all DELETES.

3301 NOTE The oic.r.cred Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in  
3302 ISO/IEC 30118-1:2018.

3303 "oic.r.cred" Resource is defined in Table 28.

3304 **Table 28 – Definition of the oic.r.cred Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/cred	Credentials	oic.r.cred	baseline	Resource containing credentials for Device authentication, verification and data protection	Security

3305 **Table 29** defines the Properties of the "/oic/sec/cred" Resource.

Table 29 – Properties of the /oic/sec/cred Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
Credentials	creds	oic.sec.cred	array	Yes	RESET	R	Server shall set to manufacturer defaults.
					RFOTM	RW	Set by DOTS after successful OTM
					RFPRO	RW	Set by the CMS (referenced via the rowneruuid Property of /oic/sec/cred Resource) after successful authentication. Access to NCRs is prohibited.
					RFNOP	R	Access to NCRs is permitted after a matching ACE is found.
					SRESET	RW	The DOTS (referenced via devowneruuid Property of /oic/sec/doxm Resource or the rowneruuid Property of /oic/sec/doxm Resource) should evaluate the integrity of and may update creds entries when a secure session is established and the Server and DOTS are authenticated.
Resource Owner ID	rowneruuid	String	uuid	Yes	RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
					RFOTM	RW	The DOTS shall configure the rowneruuid Property of /oic/sec/cred Resource when a successful owner transfer session is established.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	RW	The DOTS (referenced via devowneruuid Property of /oic/sec/doxm Resource or the rowneruuid Property of /oic/sec/doxm Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET Device state.

3307 All secure Device accesses shall have a /oic/sec/cred Resource that protects the end-to-end interaction.

3309 The /oic/sec/cred Resource shall be updateable by the service named in it's rowneruuid Property.

3310 ACLs naming /oic/sec/cred Resource should further restrict access beyond CRUDN access modes.

3311 Table 30 defines the Properties of "oic.sec.cred".

Table 30 – Properties of the oic.sec.cred Property

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Credential ID	credid	UINT16	0 – 64K-1	Yes	RW		Short credential ID for local references from other Resource
Subject UUID	subjectuuid	String	uuid	Yes	RW		A uuid that identifies the subject to which this credential applies or "" if any identity is acceptable
Role ID	roleid	oic.sec.roletype	-	No	RW		Identifies the role(s) the subject is authorized to assert.
Credential Type	credtype	oic.sec.credtype	bitmask	Yes	RW		Represents this credential's type. 0 – Used for testing 1 – Symmetric pair-wise key 2 – Symmetric group key 4 – Asymmetric signing key 8 – Asymmetric signing key with certificate 16 – PIN or password 32 – Asymmetric encryption key
Credential Usage	credusage	oic.sec.credusage	String	No	RW		Used to resolve undecidability of the credential. Provides indication for how/where the cred is used oic.sec.cred.trustca: certificate trust anchor oic.sec.cred.cert: identity certificate oic.sec.cred.rolecert: role certificate oic.sec.cred.mfgtrustca: manufacturer certificate trust anchor oic.sec.cred.mfgcert: manufacturer certificate
Public Data	publicdata	oic.sec.pubdatatype	-	No	RW		Public credential information 1:2: ticket, public SKDC values 4, 32: Public key value 8: A chain of one or more certificate
Private Data	privatedata	oic.sec.privdatatype	-	No	-	RESET	Server shall set to manufacturer default
					RW	RFOTM	Set by DOTS after successful OTM
					W	RFPRO	Set by authenticated DOTS or CMS
					-	RFNOP	Not writable during normal operation.
W	SRESET	DOTS may modify to enable transition to RFPRO.					
Optional Data	optionaldata	oic.sec.optdatatype	-	No	RW		Credential revocation status information 1, 2, 4, 32: revocation status information 8: Revocation information
Period	period	String	-	No	RW		Period as defined by IETF RFC 5545. The credential should not be used if the current time is outside the Period window.
Credential Refresh Method	crms	oic.sec.crmtype	array	No	RW		Credentials with a Period Property are refreshed using the credential refresh method (crm) according to the type definitions for oic.sec.crm.

3313 Table 31 defines the Properties of "oic.sec.credusagetype".

3314 **Table 31: Properties of the oic.sec.credusagetype Property**

Value Type Name	Value Type URN (mandatory)
Trust Anchor	oic.sec.cred.trustca
Certificate	oic.sec.cred.cert
Role Certificate	oic.sec.cred.rolecert
Manufacturer Trust CA	oic.sec.cred.mfgtrustca
Manufacturer CA	oic.sec.cred.mfgcert

3315 Table 32 defines the Properties of "oic.sec.pubdatatype".

3316 **Table 32 – Properties of the oic.sec.pubdatatype Property**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	N/A	RW	No	A string specifying the encoding format of the data contained in the pubdata "oic.sec.encoding.jwt" - IETF RFC 7519 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - IETF RFC 8392 CBOR web token (CWT) encoding "oic.sec.encoding.base64" – Base64 encoding "oic.sec.encoding.uri" – URI reference "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate or chain "oic.sec.encoding.der" – Encoding for DER-encoded certificate or chain "oic.sec.encoding.raw" – Raw hex encoded data
Data	data	String	N/A	RW	No	The encoded value

3317 Table 33 defines the Properties of "oic.sec.privdatatype".

3318 **Table 33 – Properties of the oic.sec.privdatatype Property**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Encoding format	encoding	String	N/A	RW	Yes	A string specifying the encoding format of the data contained in the privdata "oic.sec.encoding.jwt" - IETF RFC 7519 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - IETF RFC 8392 CBOR web token (CWT) encoding "oic.sec.encoding.base64" – Base64 encoding "oic.sec.encoding.uri" – URI reference "oic.sec.encoding.handle" – Data is contained in a storage sub-system referenced using a handle "oic.sec.encoding.raw" – Raw hex encoded data
Data	data	String	N/A	W	No	The encoded value This value shall not be RETRIEVE-able.
Handle	handle	UINT16	N/A	RW	No	Handle to a key storage resource

3319 Table 34 defines the Properties of "oic.sec.optdatatype".

3320

**Table 34 – Properties of the oic.sec.optdatatype Property**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Revocation status	revstat	Boolean	T   F	RW	Yes	Revocation status flag True – revoked False – not revoked
Encoding format	encoding	String	N/A	RW	No	A string specifying the encoding format of the data contained in the optdata "oic.sec.encoding.jwt" – IETF RFC 7519 JSON web token (JWT) encoding "oic.sec.encoding.cwt" - IETF RFC 8392 CBOR web token (CWT) encoding "oic.sec.encoding.base64" – Base64 encoding "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate or chain "oic.sec.encoding.der" – Encoding for DER-encoded certificate or chain "oic.sec.encoding.raw" – Raw hex encoded data
Data	data	String	N/A	RW	No	The encoded structure

3321 Table 35 defines the Properties of "oic.sec.roletype".

3322

**Table 35 – Definition of the oic.sec.roletype Property.**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Authority	authority	String	N/A	R	No	A name for the authority that defined the role. If not present, the credential issuer defined the role. If present, must be expressible as an ASN.1 PrintableString.
Role	role	String	N/A -	R	Yes	An identifier for the role. Must be expressible as an ASN.1 PrintableString.

3323 **13.3.2 Properties of the Credential Resource**

3324 **13.3.2.1 Credential ID**

3325 Credential ID (credid) is a local reference to an entry in a creds Property array of the /oic/sec/cred Resource. The SRM generates it. The credid Property shall be used to disambiguate array elements of the creds Property.

3328 **13.3.2.2 Subject UUID**

3329 The subjectuid Property identifies the Device to which an entry in a creds Property array of the /oic/sec/cred Resource shall be used to establish a secure session, verify an authentication challenge-response or to authenticate an authentication challenge.

3332 A subjectuid Property that matches the Server's own deviceuuid Property, distinguishes the array entries in the creds Property that pertain to this Device.

3334 The subjectuid Property shall be used to identify a group to which a group key is used to protect shared data.

3336 When certificate chain is used during secure connection establishment, the "subjectuuid" Property  
3337 shall also be used to verify the identity of the responder. The presented certificate chain shall be  
3338 accepted, if there is a matching Credential entry on the Device that satisfies all of the following:

- 3339 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3340 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity certificate  
3341 in the presented chain. If Subject UUID of the entry is set as a wildcard "\*", this condition is  
3342 automatically satisfied.
- 3343 – Credential Usage of the entry is "oic.sec.cred.trustca".

#### 3344 **13.3.2.3 Role ID**

3345 The roleid Property identifies a role that has been granted to the credential.

#### 3346 **13.3.2.4 Credential Type**

3347 The credtype Property is used to interpret several of the other Property values whose contents can  
3348 differ depending on credential type. These Properties include publicdata, privatedata and  
3349 optionaldata. The credtype Property value of '0' ("no security mode") is reserved for testing and  
3350 debugging circumstances. Production deployments shall not allow provisioning of credentials of  
3351 type '0'. The SRM should introduce checking code that prevents its use in production deployments.

#### 3352 **13.3.2.5 Public Data**

3353 The publicdata Property contains information that provides additional context surrounding the  
3354 issuance of the credential. For example, it might contain information included in a certificate or  
3355 response data from a CMS. It might contain wrapped data.

#### 3356 **13.3.2.6 Private Data**

3357 The privatedata Property contains secret information that is used to authenticate a Device, protect  
3358 data or verify an authentication challenge-response.

3359 The privatedata Property shall not be disclosed outside of the SRM's trusted computing perimeter.  
3360 A secure element (SE) or trusted execution environment (TEE) should be used to implement the  
3361 SRM's trusted computing perimeter. The privatedata contents may be referenced using a handle;  
3362 for example if used with a secure storage sub-system.

#### 3363 **13.3.2.7 Optional Data**

3364 The optionaldata Property contains information that is optionally supplied, but facilitates key  
3365 management, scalability or performance optimization.

#### 3366 **13.3.2.8 Period**

3367 The period Property identifies the validity period for the credential. If no validity period is specified  
3368 the credential lifetime is undetermined. Constrained devices that do not implement a date-time  
3369 capability shall obtain current date-time information from its CMS.

#### 3370 **13.3.2.9 Credential Refresh Method Type Definition**

3371 The CMS shall implement the credential refresh methods specified in the crms Property of the  
3372 oic.sec.creds array in the /oic/sec/cred Resource.

3373 Table 36 defines the values of "oic.sec.crmtype".

Table 36 – Value Definition of the oic.sec.crmtyp Property

Value Type Name	Value Type URN	Applicable Credential Type	Description
Provisioning Service	oic.sec.crm.pro	All	A CMS initiates re-issuance of credentials nearing expiration. The Server should delete expired credentials to manage storage resources. The Resource Owner Property references the provisioning service. The Server uses its /oic/sec/cred.rowneruoid Resource to identify additional key management service that supports this credential refresh method.
Pre-shared Key	oic.sec.crm.psk	[1]	The Server performs ad-hoc key refresh by initiating a DTLS connection with the Device prior to credential expiration using a Diffie-Hellman based ciphersuite and the current PSK. The new DTLS MasterSecret value becomes the new PSK. The Server selects the new validity period. The new validity period value is sent to the Device who updates the validity period for the current credential. The Device acknowledges this update by returning a successful response or denies the update by returning a failure response. The Server uses its /oic/sec/cred.rowneruoid Resource to identify a key management service that supports this credential refresh method.
Random PIN	oic.sec.crm.rdp	[16]	The Server performs ad-hoc key refresh following the oic.sec.crm.psk approach, but in addition generates a random PIN value that is communicated out-of-band to the remote Device. The current PSK + PIN are hashed to form a new PSK' that is used with the DTLS ciphersuite. I.e. PSK' = SHA256(PSK, PIN). The Server uses its /oic/sec/cred.rowneruoid Resource to identify a key management service that supports this credential refresh method.
SKDC	oic.sec.crm.skdc	[1, 2, 4, 32]	The Server issues a request to obtain a ticket for the Device. The Server updates the credential using the information contained in the response to the ticket request. The Server uses its /oic/sec/cred.rowneruoid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruoid Resource to identify a key management service that supports this credential refresh method.
PKCS10	oic.sec.crm.pk10	[8]	The Server issues a PKCS#10 certificate request message to obtain a new certificate. The Server uses its /oic/sec/cred.rowneruoid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruoid Resource to identify a key management service that supports this credential refresh method.

3375 **13.3.2.10 Credential Usage**

3376 Credential Usage indicates to the Device the circumstances in which a credential should be used.  
3377 Five values are defined:

- 3378 – oic.sec.cred.trustca: This certificate is a trust anchor for the purposes of certificate chain  
3379 validation, as defined in 10.3.
- 3380 – oic.sec.cred.cert: This credusage is used for certificates for which the Device possesses the  
3381 private key and uses it for identity authentication in a secure session, as defined in clause 10.4.
- 3382 – oic.sec.cred.rolecert: This credusage is used for certificates for which the Device possesses  
3383 the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3384 – oic.sec.cred.mfgtrustca: This certificate is a trust anchor for the purposes of the Manufacturer  
3385 Certificate Based OTM as defined in clause 7.3.6.
- 3386 – oic.sec.cred.mfgcert: This certificate is used for certificates for which the Device possesses the  
3387 private key and uses it for authentication in the Manufacturer Certificate Based OTM as defined  
3388 in clause 7.3.6.

3389 **13.3.3 Key Formatting**

3390 **13.3.3.1 Symmetric Key Formatting**

3391 Symmetric keys shall have the format described in Table 37 and Table 38.

3392 **Table 37 – 128-bit symmetric key**

Name	Value	Type	Description
Length	16	OCTET	Specifies the number of 8-bit octets following Length
Key	opaque	OCTET Array	16 byte array of octets. When used as input to a PSK function Length is omitted.

3393

3394 **Table 38 – 256-bit symmetric key**

Name	Value	Type	Description
Length	32	OCTET	Specifies the number of 8-bit octets following Length
Key	opaque	OCTET Array	32 byte array of octets. When used as input to a PSK function Length is omitted.

3395 **13.3.3.2 Asymmetric Keys**

3396 Asymmetric key formatting is not available in this revision of the document.

3397 **13.3.3.3 Asymmetric Keys with Certificate**

3398 Key formatting is defined by certificate definition.

3399 **13.3.3.4 Passwords**

3400 Password formatting is not available in this revision of the document.

3401 **13.3.4 Credential Refresh Method Details**

3402 **13.3.4.1 Provisioning Service**

3403 The resource owner identifies the provisioning service. If the Server determines a credential  
3404 requires refresh and the other methods do not apply or fail, the Server will request re-provisioning  
3405 of the credential before expiration. If the credential is allowed to expire, the Server should delete  
3406 the Resource.

3407 **13.3.4.2 Pre-Shared Key**

3408 **13.3.4.2.1 Pre-Shared Key General**

3409 Using this mode, the current PSK is used to establish a Diffie-Hellmen session key in DTLS. The  
3410 TLS\_PRf is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3411  $PSK = TLS\_PRF(MasterSecret, Message, length);$

- 3412 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of the
- 3413 above ciphersuites.
- 3414 – Message is the concatenation of the following values:
  - 3415 – RM - Refresh method – I.e. "oic.sec.crm.psk"
  - 3416 – Device ID\_A is the string representation of the Device ID that supplied the DTLS ClientHello.
  - 3417 – Device ID\_B is the Device responding to the DTLS ClientHello message
- 3418 – Length of Message in bytes.

3419 Both Server and Client use the PSK to update the /oic/sec/cred Resource's privatedata Property.  
3420 If Server initiated the credential refresh, it selects the new validity period. The Server sends the  
3421 chosen validity period to the Client over the newly established DTLS session so it can update the  
3422 corresponding credential Resource for the Server.

#### 3423 **13.3.4.2.2 Random PIN**

3424 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.  
3425 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session  
3426 key should be used to switch from PIN to PSK mode.

3427 The PIN is randomly generated by the Server and communicated to the Client through an out-of-  
3428 band method. The OOB method used is out-of-scope.

3429 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used to  
3430 generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a DTLS  
3431 ciphersuite that accepts a PSK.

3432  $PPSK = PBKDF2(PRF, PIN, RM, Device\ ID, c, dkLen)$

3433 The PBKDF2 function has the following parameters:

- 3434 – PRF – Uses the DTLS PRF.
- 3435 – PIN – Shared between Devices.
- 3436 – RM - Refresh method – I.e. "oic.sec.crm.rdp"
- 3437 – Device ID – UUID of the new Device.
- 3438 – c – Iteration count initialized to 1000, incremented upon each use.
- 3439 – dkLen – Desired length of the derived PSK in octets.

3440 Both Server and Client use the PPSK to update the /oic/sec/cred Resource's PrivateData Property.  
3441 If Server initiated the credential refresh, it selects the new validity period. The Server sends the  
3442 chosen validity period to the Client over the newly established DTLS session so it can update its  
3443 corresponding credential Resource for the Server.

#### 3444 **13.3.4.2.3 SKDC**

3445 A DTLS session is opened to the Server where the /oic/sec/cred Resource has an rowneruid  
3446 Property value that matches the a CMS that implements SKDC functionality and where the Client  
3447 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request message  
3448 is delivered to the CMS and in response returns the ticket request. The Server updates or  
3449 instantiates an /oic/sec/cred Resource guided by the ticket response contents.

#### 3450 **13.3.4.2.4 PKCS10**

3451 A DTLS session is opened to the Server where the /oic/sec/cred Resource has an rowneruid  
3452 Property value that matches the a CMS that supports the oic.sec.crm.pk10 credential refresh  
3453 method. A PKCS10 formatted message is delivered to the service. After the refreshed certificate is  
3454 issued, the CMS pushes the certificate to the Server. The Server updates or instantiates an  
3455 /oic/sec/cred Resource guided by the certificate contents.

#### 3456 **13.3.4.3 Resource Owner**

3457 The Resource Owner Property allows credential provisioning to occur soon after Device onboarding  
3458 before access to support services has been established. It identifies the entity authorized to  
3459 manage the /oic/sec/cred Resource in response to Device recovery situations.

3460 **13.4 Certificate Revocation List**

3461 **13.4.1 CRL Resource Definition**

3462 Device certificates and private keys are kept in `cred` Resource. CRL is maintained and updated  
 3463 with a separate `crl` Resource that is newly defined for maintaining the revocation list.

3464 "oic.r.crl" Resource is defined in Table 39.

3465 **Table 39 – Definition of the oic.r.crl Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/crl	CRLs	oic.r.crl	baseline	Resource containing CRLs for Device certificate revocation	Security

3466 Table 40 defines the Properties of "oic.r.crl".

3467 **Table 40 – Properties of the oic.r.crl Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
CRL Id	crlid	UINT16	0 – 64K-1	RW	Yes	CRL ID for references from other Resource
This Update	thisupdate	String	N/A	RW	Yes	This indicates the time when this CRL has been updated.(UTC)
CRL Data	crldata	String	N/A	RW	Yes	CRL data based on CertificateList in CRL profile

3468 **13.5 ACL Resources**

3469 **13.5.1 ACL Resources General**

3470 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be  
 3471 expressed using three ACL Resource Types: /oic/sec/acl2, /oic/sec/amacl and /oic/sec/sacl. The  
 3472 subject (e.g. deviceuid of the Client) requesting access to a Resource shall be authenticated prior  
 3473 to applying the ACL check. Resources that are available to multiple Clients can be matched using  
 3474 a wildcard subject. All Resources accessible via the unsecured communication endpoint shall be  
 3475 matched using a wildcard subject.

3476 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3477 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3478 **Table 41 – BNF Definition of OCF ACL**

<ACL>	<ACE> {<ACE>}
<ACE>	<SubjectId> <ResourceRef> <Permission> {<Validity>}
<SubjectId>	<DeviceId>   <Wildcard>   <RoleId>
<DeviceId>	<UUID>
<RoleId>	<Character>   <RoleName><Character>
<RoleName>	"*"   <Authority><Character>
<Authority>	<UUID>
<ResourceRef>	' (' <OIC_LINK> {',' {OIC_LINK}> ')'
<Permission>	('C'   '-' ) ('R'   '-' ) ('U'   '-' ) ('D'   '-' ) ('N'   '-' )
<Validity>	<Period> {<Recurrence>}

<Wildcard>	'**'
<URI>	IETF RFC 3986
<UUID>	IETF RFC 4122
<Period>	IETF RFC 5545 Period
<Recurrence>	IETF RFC 5545 Recurrence
<OIC_LINK>	ISO/IEC 30118-1:2018 defined in JSON Schema
<Character>	<Any UTF8 printable character, excluding NUL>

- 3479 The <Deviceld> token means the requestor must possess a credential that uses <UUID> as its  
3480 identity in order to match the requestor to the <ACE> policy.
- 3481 The <RoleID> token means the requestor must possess a role credential with <Character> as its  
3482 role in order to match the requestor to the <ACE> policy.
- 3483 The <Wildcard> token "\*" means any requestor is matched to the <ACE> policy, with or without  
3484 authentication.
- 3485 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the <ACE>  
3486 policy to Resources.
- 3487 The <OIC\_LINK> token contains values used to query existence of hosted Resources.
- 3488 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>  
3489 and <ResourceRef> matching does not produce the empty set match.
- 3490 Permissions are defined in terms of CREATE ('C'), RETRIEVE ('R'), UPDATE ('U'), DELETE ('D'),  
3491 NOTIFY ('N') and NIL ('-'). NIL is substituted for a permissions character that signifies the  
3492 respective permission is not granted.
- 3493 The empty set match result defaults to a condition where no access rights are granted.
- 3494 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.  
3495 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively  
3496 be granted and rescinded according to the pattern.

### 3497 13.5.3 ACL Resource

3498 ~~There are two types of ACLs, 'acl' is a list of type 'ace' and An 'acl2' is a list of type 'ace2'. A Device~~  
3499 ~~shall not host the /acl Resource.~~

3500 ~~NOTE—The /acl Resource is defined for backward compatibility and use by Provisioning Tools, etc.~~

3501 In order to provide an interface which allows management of array elements of the "aclist2"  
3502 Property associated with an /oic/sec/acl2 Resource. The RETRIEVE, UPDATE and DELETE  
3503 operations on the /oic/sec/acl2 Resource SHALL behave as follows:

- 3504 1) A RETRIEVE shall return the full Resource representation.
- 3505 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3506 UPDATE request, as follows:
  - 3507 a) If an UPDATE representation includes the array Property, then:
    - 3508 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace completely  
3509 the corresponding ACE in the existing "aces2" array.
    - 3510 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array, and  
3511 a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new ACE  
3512 by the Server. The "aceid" of a deleted ACE should not be reused, to improve the  
3513 determinism of the interface and reduce opportunity for race conditions.

3514 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be  
3515 appended to the existing "aces2" array, using the supplied "aceid".

3516 The rows in Table 47 defines the Properties of "oic.sec.acl2".

3517 iv) Table 47 corresponding to the "aclist2" array Property dictate the Device States in which  
3518 an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is in a  
3519 Device State where the Access Mode in this row contains "R", then the OCF Device  
3520 shall reject all UPDATES of the "aclist2" array Property.

3521 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not remove  
3522 the oic.r.ace2 Resource.

3523 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the  
3524 corresponding aceid(s) from the "aces2" array.

3525 The rows in Table 47 defines the Properties of "oic.sec.acl2".

3526 5) Table 47 corresponding to the "aclist2" array Property dictate the Device States in which a  
3527 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this  
3528 row contains "R", then the OCF Device shall reject all DELETES.

3529 NOTE The oic.r.acl2 Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3530 in ISO/IEC 30118-1:2018.

3531 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no  
3532 entry can be found for the requested Resource for the requestor – e.g. /oic/sec/acl2, /oic/sec/sacl  
3533 and /oic/sec/amacl do not match the subject and the requested Resource.

3534 It is possible the AMS has an ACL policy that satisfies a resource access request, but the necessary  
3535 ACE has not been provisioned to Server. The Server may open a secure connection to the AMS to  
3536 request ACL provisioning. The Server may use filter criteria that returns a subset of the AMS ACL  
3537 policy. The AMS shall obtain the Server Device ID using the secure connection context.

3538 The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS to  
3539 process an /oic/sec/amacl Resource. The AMS shall match the AMACL policy and return the  
3540 Permission Property or an error if no match is found.

3541 If the requested Resource is still not matched, the Server returns an error. The requester should  
3542 query the Server to discover the configured AMS services. The Client should contact the AMS to  
3543 request a sacl (/oic/sec/sacl) Resource. Performing the following operations implement this type of  
3544 request:

3545 1) Client: Open secure connection to AMS.

3546 2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3547 3) AMS: constructs a /oic/sec/sacl Resource that is signed by the AMS and returns it in response  
3548 to the RETRIEVE command.

3549 4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3550 5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3551 6) Client: retries original Resource access request. This time the new ACL is included in the local  
3552 ACL evaluation.

3553 The ACL contained in the /oic/sec/sacl Resource should grant longer term access that satisfies  
3554 repeated Resource requests.

3555 ~~"oic.r.acl" Resource is defined in Table 42.~~

3556 ~~Table 42 – Definition of the oic.r.acl Resource~~

Formatted: PARAGRAPH

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/acl	ACL	oic.r.acl	baseline	Resource for managing access	Security

Formatted: PARAGRAPH

3557 Table 43 defines the Properties of "oic.r.acl".

3558 Table 43 – Properties of the oic.r.acl Resource

Formatted: PARAGRAPH

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
ACE List	aclist	oic.sec.ace	N/A	Yes	N/A	N/A	Access Control Entries in the ACL resource. This Property contains "aces", an array of oic.sec.ace1 resources and "aces2", an array of oic.sec.ace2 Resources
N/A	N/A	N/A	N/A	N/A	R	RESET	Server shall set to manufacturer defaults.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
					R	RFNOP	Access to NCRs is permitted after a matching ACE is found.
					RW	SRESET	The DOTS (referenced via devowneruuid Property of /oic/sec/dotm Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
Resource Owner ID	rowneruid	String	uuid	Yes	N/A	N/A	The resource owner Property (rowneruid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
					R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOTS should configure the /acl/rowneruid Property when a successful owner transfer session is established.

Formatted: PARAGRAPH

Formatted: PARAGRAPH

Formatted: PARAGRAPH

					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via /doxm-devowneruuid Property or the /doxm-rownruid Property) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3559 Table 44 defines the Properties of "oic.r.ace".

3560

**Table 44 – Properties of the oic.r.ace Property**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Resources	resources	oic.oic-link	array	RW	Yes	The application's Resources to which a security policy applies
Permission	permission	oic.sec.crudn type	bitmask	RW	Yes	Bitmask encoding of CRUDN permission
Validity	validity	oic.sec.ace/d efinitions/tim e-interval	array	RW	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the IETF RFC 5545 Period, and a string array representing a recurrence rule using the IETF RFC 5545 Recurrence.
Subject ID	subjectuuid	String	uuid, ""	RW	Yes	A uuid that identifies the Device to which this ACE applies to or "" for anonymous access.

3561 Table 45 defines the values of "oic.sec.crudntype".

3562

**Table 42 – Value Definition of the oic.sec.crudntype Property**

Value	Access Policy	Description	RemarksNotes
bx0000,0000 (0)	No permissions	No permissions	N/A
bx0000,0001 (1)	C	CREATE	N/A
bx0000,0010 (2)	R	RETREIVE, OBSERVE, DISCOVER	The "R" permission bit covers both the Read permission and the Observe permission.
bx0000,0100 (4)	U	WRITE, UPDATE	N/A
bx0000,1000 (8)	D	DELETE	N/A
bx0001,0000 (16)	N	NOTIFY	The "N" permission bit is ignored in OCF 1.0, since "R" covers the Observe permission. It is documented for future versions

3563 "oic.sec.acl2" Resourcce is defined in Table 28.

3564

**Table 43 – Definition of the oic.sec.acl2 Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/acl2	ACL2	oic.r.acl2	baseline	Resource for managing access	Security

3565 Table 47 defines the Properties of "oic.sec.acl2".

**Table 44 – Properties of the oic.sec.acl2 Resource**

Property Name	Value Type	Mandatory	Device State	Access Mode	Description
aclist2	array of oic.sec.ace2	Yes	N/A		The aclist2 Property is an array of ACE records of type "oic.sec.ace2". The Server uses this list to apply access control to its local resources.
N/A	N/A	N/A	RESET	R	Server shall set to manufacturer defaults.
			RFOTM	RW	Set by DOTS after successful OTM
			RFPRO	RW	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
			RFNOP	R	Access to NCRs is permitted after a matching ACE2 is found.
			SRESET	RW	The DOTS (referenced via devowneruuid Property of /oic/sec/doxm Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
rowneruuid	uuid	Yes	N/A		The resource owner Property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
			RFOTM	RW	The DOTS should configure the rowneruuid Property of /oic/sec/acl2 Resource when a successful owner transfer session is established.
			RFPRO	R	n/a
			RFNOP	R	n/a
			SRESET	RW	The DOTS (referenced via devowneruuid Property or rowneruuid Property of /oic/sec/doxm Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3569

**Table 45 – oic.sec.ace2 data type definition.**

Property Name	Value Type	Mandatory	Description
subject	oic.sec.roletype, oic.sec.didtype, oic.sec.conntype	Yes	The Client is the subject of the ACE when the roles, Device ID, or connection type matches.
resources	array of oic.sec.ace2.resource- ref	Yes	The application's resources to which a security policy applies
permission	oic.sec.crudntype.bitm- ask	Yes	Bitmask encoding of CRUDN permission
validity	array of oic.sec.time- pattern	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the IETF RFC 5545 Period, and a string array representing a recurrence rule using the IETF RFC 5545 Recurrence.
aceid	integer	Yes	An aceid is unique with respect to the array entries in the aclist2 Property.

3570 Table 49 defines the Properties of "oic.sec.ace2.resource-ref".

3571

**Table 46 – oic.sec.ace2.resource-ref data type definition.**

Property Name	Value Type	Mandatory	Description
href	uri	No	A URI referring to a resource to which the containing ACE applies
wc	string	No	Refer to Table 23.

3572 Table 50 defines the values of "oic.sec.ace2.resource-ref".

3573

**Table 47 – Value definition oic.sec.conntype Property**

Property Name	Value Type	Value Rule	Description
conntype	string	enum [ "auth-crypt", "anon-clear" ]	This Property allows an ACE to be matched based on the connection or message protection type
		auth-crypt	ACE applies if the Client is authenticated and the data channel or message is encrypted and integrity protected
		anon-clear	ACE applies if the Client is not authenticated and the data channel or message is not encrypted but may be integrity protected

3574 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack  
 3575 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's  
 3576 request using policies contained in ACL resources.

3577 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified  
 3578 Resource references include the device identifier in the href Property that identifies the remote  
 3579 Resource Server that hosts the Resource. Partially qualified references means the local Resource  
 3580 Server hosts the Resource. If a fully qualified resource reference is given, the Intermediary  
 3581 enforcing access shall have a secure channel to the Resource Server and the Resource Server  
 3582 shall verify the Intermediary is authorized to act on its behalf as a Resource access enforcement  
 3583 point.

3584 Resource Servers should include references to Device and ACL Resources where access  
3585 enforcement is to be applied. However, access enforcement logic shall not depend on these  
3586 references for access control processing as access to Server Resources will have already been  
3587 granted.

3588 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and modify  
3589 this Resource. This prevents non-terminating dependency on some other ACL Resource.  
3590 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL  
3591 Resource.

3592 An ~~ACE-or~~ ACE2 entry is considered called "currently valid" if the validity period of the ~~ACE-or~~  
3593 ACE2 entry includes the time of the request. The validity period in the ~~ACE-or~~ ACE2 may be a  
3594 recurring time period (e.g., daily from 1:00-2:00). Matching the resource(s) specified in a request  
3595 to the resource Property of the ~~ACE-or~~ ACE2 is defined in Clause 12.2. For example, one way  
3596 they can match is if the Resource URI in the request exactly matches one of the resource  
3597 references in the ~~ACE-or~~ ACE2 entries.

3598 ~~A request will match an ACE if any of the following are true:~~

3599 ~~1) The deviceuuid Property associated with the secure session matches the "subjectuid" of the~~  
3600 ~~ACE; AND the Resource of the request matches one of the resources Property of the ACE; AND~~  
3601 ~~the ACE is currently valid.~~

3602 ~~2) The ACE subjectuid Property contains the wildcard "\*" character; AND the Resource of the~~  
3603 ~~request matches one of the resources Property of the ACE; AND the ACE is currently valid.~~

3604 ~~3) When authentication uses a symmetric key credential;~~

3605 ~~AND the CoAP payload query string of the request specifies a role, which is associated with~~  
3606 ~~the symmetric key credential of the current secure session;~~

3607 ~~AND the CoAP payload query string of the request specifies a role, which is contained in the~~  
3608 ~~oic.r.creds.roleid Property of the current secure session;~~

3609 ~~AND the resource of the request matches one of the resources Property of the ACE;~~

3610 ~~AND the ACE is currently valid.~~

3611 A request will match an ACE2 if any of the following are true:

3612 1) The ACE2 subject Property is of type oic.sec.didtype has a UUID value that matches the  
3613 deviceuuid Property associated with the secure session;

3614 AND the Resource of the request matches one of the resources Property of the ACE2  
3615 oic.sec.ace2.resource-ref;

3616 AND the ACE2 is currently valid.

3617 2) The ACE2 subject Property is of type oic.sec.conntype and has the wildcard value that matches  
3618 the currently established connection type;

3619 AND the resource of the request matches one of the resources Property of the ACE2  
3620 oic.sec.ace2.resource-ref;

3621 AND the ACE2 is currently valid.

3622 3) When Client authentication uses a certificate credential;

3623 AND one of the roleid values contained in the role certificate matches the roleid Property of the  
3624 ACE2 oic.sec.roletype;

3625 AND the role certificate public key matches the public key of the certificate used to establish  
3626 the current secure session;

3627 AND the resource of the request matches one of the array elements of the resources Property  
3628 of the ACE2 oic.sec.ace2.resource-ref;

3629 AND the ACE2 is currently valid.

3630 4) When Client authentication uses a certificate credential;

3631 AND the CoAP payload query string of the request specifies a role, which is member of the set

3632 of roles contained in the role certificate;

3633 AND the roleid values contained in the role certificate matches the roleid Property of the ACE2

3634 oic.sec.roletype;

3635 AND the role certificate public key matches the public key of the certificate used to establish

3636 the current secure session;

3637 AND the resource of the request matches one of the resources Property of the ACE2

3638 oic.sec.ace2.resource-ref;

3639 AND the ACE2 is currently valid.

3640 5) When Client authentication uses a symmetric key credential;

3641 AND one of the roleid values associated with the symmetric key credential used in the secure

3642 session, matches the roleid Property of the ACE2 oic.sec.roletype;

3643 AND the resource of the request matches one of the array elements of the resources Property

3644 of the ACE2 oic.sec.ace2.resource-ref;

3645 AND the ACE2 is currently valid.

3646 6) When Client authentication uses a symmetric key credential;

3647 AND the CoAP payload query string of the request specifies a role, which is contained in the

3648 oic.r.cred.creds.roleid Property of the current secure session;

3649 AND CoAP payload query string of the request specifies a role that matches the roleid Property

3650 of the ACE2 oic.sec.roletype;

3651 AND the resource of the request matches one of the array elements of the resources Property

3652 of the ACE2 oic.sec.ace2.resource-ref;

3653 AND the ACE2 is currently valid.

3654 A request is granted if ANY of the 'matching' ACE2s [entries](#) contains the permission to allow the

3655 request. Otherwise, the request is denied.

3656 There is no way for an ACE2 [entry](#) to explicitly deny permission to a resource. Therefore, if one

3657 Device with a given role should have slightly different permissions than another Device with the

3658 same role, they must be provisioned with different roles.

3659 The Server is required to verify that any hosted Resource has authorized access by the Client

3660 requesting access. The /oic/sec/acl2 Resource is co-located on the Resource host so that the

3661 Resource request processing should be applied securely and efficiently. See Annex A for example.

### 3662 13.6 Access Manager ACL Resource

3663 "oic.r.amacl" Resource is defined in Table 51.

3664 **Table 48 – Definition of the oic.r.amacl Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/amacl	Managed ACL	oic.r.amacl	baseline	Resource for managing access	Security

3665 Table 52 defines the Properties of "oic.r.amacl".

3666

**Table 49 – Properties of the oic.r.amacl Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Resources	resources	oic.sec.ace2.resource-ref	array	RW	Yes	Multiple links to this host's Resources

3667 The AMS should be used to centralize management of access policy, but requires Servers to open  
 3668 a connection to the AMS whenever the named Resources are accessed. See A.2 for example.

### 3669 13.7 Signed ACL Resource

3670 "oic.r.sacl" Resource is defined in Table 53.

3671

**Table 50 – Definition of the oic.r.sacl Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/sacl	Signed ACL	oic.r.sacl	baseline	Resource for managing access	Security

3672 Table 54 defines the Properties of "oic.r.sacl".

3673

**Table 51 – Properties of the oic.r.sacl Resource**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	State	Description
ACE List	aclist2	oic.sec.ace2	array	Yes	N/A	N/A	Access Control Entries in the ACL Resource
					N/A	RESET	Server shall set to manufacturer defaults.
					N/A	RFOTM	Set by DOTS after successful OTM
					N/A	RFPRO	The AMS (referenced via rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
					N/A	RFNOP	Access to NCRs is permitted after a matching ACE is found.
					N/A	SRESET	The DOTS (referenced via devowneruuid Property of /oic/sec/doxm Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
Signature	signature	oic.sec.sigtype	N/A	Yes	N/A	N/A	The signature over the ACL Resource

3674 Table 55 defines the Properties of "oic.sec.sigtype".

3675

**Table 52 – Properties of the oic.sec.sigtype Property**

Property Title	Property Name	Value Type	Value Rule	Unit	Access Mode	Mandatory	Description
Signature Type	sigtype	String	N/A	N/A	RW	Yes	The string specifying the predefined signature format. "oic.sec.sigtype.jws" – IETF RFC 7515 JSON web signature (JWS) object "oic.sec.sigtype.pk7" – IETF RFC 2315 base64-encoded object "oic.sec.sigtype.cws" – CBOR-encoded JWS object
Signature Value	sigvalue	String	N/A	N/A	RW	Yes	The encoded signature

3676

**13.8 Provisioning Status Resource**

3677 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning should  
 3678 be Client-directed or Server-directed. Client-directed provisioning relies on a Client device to  
 3679 determine what, how and when Server Resources should be instantiated and updated. Server-  
 3680 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-  
 3681 directed provisioning depends on configuration of the rowneruid Property of the /oic/sec/doxm,  
 3682 /oic/sec/cred and /oic/sec/acl2 Resources to identify the device ID of the trusted DOTS, CMS and  
 3683 AMS services respectively. Furthermore, the /oic/sec/cred Resource should be provisioned at  
 3684 ownership transfer with credentials necessary to open a secure connection with appropriate  
 3685 support service.

3686

"oic.r.pstat" Resource is defined in Table 56.

3687

**Table 53 – Definition of the oic.r.pstat Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/pstat	Provisioning Status	oic.r.pstat	baseline	Resource for managing Device provisioning status	Configuration

3688

Table 57 defines the Properties of "oic.r.pstat".

Table 54 – Properties of the oic.r.pstat Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	dos	oic.sec.dostype	N/A	Yes	RW		Device Onboarding State
Is Device Operational	isop	Boolean	T F	Yes	R	RESET	Server shall set to FALSE
					R	RFOTM	Server shall set to FALSE
					R	RFPRO	Server shall set to FALSE
					R	RFNOP	Server shall set to TRUE
					R	SRESET	Server shall set to FALSE
Current Mode	cm	oic.sec.dpmttype	bitmask	Yes			Deprecated
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes			Deprecated
Operational Mode	om	oic.sec.pomttype	bitmask	Yes	R	RESET	Server shall set to manufacturer default.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by DOTS.
Supported Mode	sm	oic.sec.pomttype	bitmask	Yes	R	All states	Supported provisioning services operation modes
Device UUID	deviceuuid	String	uuid	Yes	RW	All states	[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000" )
					RW	RFOTM	The DOTS should configure the rowneruid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruid Property of /oic/sec/doxm Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruid does not refer to a valid DOTS the Server shall transition to RESET Device state.

3690 The provisioning status Resource /oic/sec/pstat is used to enable Devices to perform self-directed  
3691 provisioning. Devices are aware of their current configuration status and a target configuration  
3692 objective. When there is a difference between current and target status, the Device should consult

3693 the rowneruuid Property of /oic/sec/cred Resource to discover whether any suitable provisioning  
 3694 services exist. The Device should request provisioning if configured to do so. The om Property of  
 3695 /oic/sec/pstat Resource will specify expected Device behaviour under these circumstances.

3696 Self-directed provisioning enables Devices to function with greater autonomy to minimize  
 3697 dependence on a central provisioning authority that should be a single point of failure in the OCF  
 3698 Security Domain.

3699 Table 58 defines the Properties of "/oic/sec/dostype".

3700 **Table 55 – Properties of the /oic/sec/dostype Property**

Property Title	Property Name	Value Type	Value Rule	Mandatory	Access Mode	Device State	Description
Device Onboarding State	s	UINT16	enum (0=RESET, 1=RFOTM, 2=RFPRO, 3=RFNOP, 4=SRESET	Y	R	RESET	The Device is in a hard reset state.
					RW	RFOTM	Set by DOTS after successful OTM to RFPRO.
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by CMS, AMS, DOTS after successful authentication
Pending state	p	Boolean	T   F	Y	R	All States	TRUE (1) – 's' state is pending until all necessary changes to Device resources are complete FALSE (0) – 's' state changes are complete

3701 In all Device states:

3702 – An authenticated and authorised Client may change the Device state of a Device by updating  
 3703 pstat.dos.s to the desired value. The allowed Device state transitions are defined in Figure 27.

3704 – Prior to updating pstat.dos.s, the Client configures the Device to meet entry conditions for the  
 3705 new Device state. The SVR definitions define the entity (Client or Server) expected to perform  
 3706 the specific SVR configuration change to meet the entry conditions. Once the Client has  
 3707 configured the aspects for which the Client is responsible, it may update pstat.dos.s. The  
 3708 Server then makes any changes for which the Server is responsible, including updating required  
 3709 SVR values, and set pstat.dos.s to the new value.

3710 – The pstat.dos.p Property is read-only by all Clients.

3711 – The Server sets pstat.dos.p to TRUE before beginning the process of updating pstat.dos.s, and  
 3712 sets it back to FALSE when the pstat.dos.s change is completed.

3713 Any requests to update pstat.dos.s while pstat.dos.p is TRUE are denied.

3714 When Device state is RESET:

3715 – All SVR content is removed and reset to manufacturer default values.

3716 – The default manufacturer Device state is RESET.

3717 – NCRs are reset to manufacturer default values.

3718 – NCRs are inaccessible.

3719 – After successfully processing RESET the SRM transitions to RFOTM by setting s Property of  
 3720 /oic/sec/dostype Resource to RFOTM.

- 3721 When Device state is RFOTM:
- 3722 – NCRs are inaccessible.
  - 3723 – Before OTM is successful, the deviceuuid Property of /oic/sec/doxm Resource shall be set to a
  - 3724 temporary non-repeated value as defined in clauses 13.2 and 13.16.
  - 3725 – Before OTM is successful, the s Property of /oic/sec/dostype Resource is read-only by
  - 3726 unauthenticated requestors
  - 3727 – After the OTM is successful, the s Property of /oic/sec/dostype Resource is read-write by
  - 3728 authorized requestors.
  - 3729 – The negotiated Device OC is used to create an authenticated session over which the DOTS
  - 3730 directs the Device state to transition to RFPRO.
  - 3731 – If an authenticated session cannot be established the ownership transfer session should be
  - 3732 disconnected and SRM sets back the Device state to RESET state.
  - 3733 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the
  - 3734 SRM asserts the OTM failed, should be disconnected, and transitions to RESET
  - 3735 (/pstat.dos.s=RESET).
  - 3736 – The DOTS UPDATES the devowneruuid Property in the /doxm Resource to a non-nil UUID
  - 3737 value. The DOTSDOXs (or other authorized client) may update it multiple times while in RFOTM.
  - 3738 It is not updatable while in other device states except when the Device state returns to RFOTM
  - 3739 through RESET.
  - 3740 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done, the
  - 3741 DOTSDOXs UPDATES the "owned" Property in the /doxm Resource to "true".
- 3742 When Device state is RFPRO:
- 3743 – The s Property of /oic/sec/dostype Resource is read-only by unauthorized requestors and read-
  - 3744 write by authorized requestors.
  - 3745 – NCRs are inaccessible, except for Easy Setup Resources, if supported.
  - 3746 – The OCF Server may re-create NCRs.
  - 3747 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
  - 3748 – An authorized Client may perform consistency checks on SVRs to determine which shall be re-
  - 3749 provisioned.
  - 3750 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if
  - 3751 the Device has already transitioned from SRESET but consistency checks continue to fail.
  - 3752 – The authorized Client sets the /pstat.dos.s=RFNOP.
- 3753 When Device state is RFNOP:
- 3754 – The /pstat.dos.s Property is read-only by unauthorized requestors and read-write by authorized
  - 3755 requestors.
  - 3756 – NCRs, SVRs and core Resources are accessible following normal access processing.
  - 3757 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or
  - 3758 RESET.
- 3759 When Device state is SRESET:
- 3760 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to
  - 3761 access or reference them.
  - 3762 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
  - 3763 include devowneruuid Property of the "/oic/sec/doxm" Resource,

3764 "creds":[...,{ "subjectuid":<devowneruid>,...}] Property of the /oic/sec/cred Resource and s  
 3765 Property of the /oic/sec/dostype Resource of /oic/sec/pstat Resource.

3766 – The certificates that identify and authorize the Device owner are sufficient to re-create  
 3767 minimalist /cred and /doxm resources enabling Device owner control of SRESET. If the SRM  
 3768 can't establish these Resources, then it will transition to RESET state.

3769 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as  
 3770 needed to ensure they are available for continued provisioning in RFPRO or for normal  
 3771 functioning in RFNOP.

3772 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING  
 3773 dos.s Property of the /pstat Resource with RFPRO or RFNOP values

3774 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to Device  
 3775 owner privileges.

3776 – The SRM asserts a Client-directed operational mode (e.g. /pstat.om=CLIENT\_DIRECTED).

3777 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.  
 3778 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning  
 3779 mode without selecting any particular value.

3780 "oic.sec.dpmttype" is defined in Table 59.

3781 **Table 56 – Definition of the oic.sec.dpmttype Property**

Type Name	Type URN	Description
Device Provisioning Mode	oic.sec.dpmttype	Device provisioning mode is a 16-bit bitmask describing various provisioning modes

3782 Table 60 and Table 61 define the values of "oic.sec.dpmttype".

3783 **Table 57 – Value Definition of the oic.sec.dpmttype Property (Low-Byte)**

Value	Device Mode	Description
bx0000,0001 (1)	Deprecated	
bx0000,0010 (2)	Deprecated	
bx0000,0100 (4)	Deprecated	
bx0000,1000 (8)	Deprecated	
bx0001,0000 (16)	Deprecated	
bx0010,0000 (32)	Deprecated	
bx0100,0000 (64)	Initiate Software Version Validation	Software version validation requested/pending (1) Software version validation complete (0)
bx1000,0000 (128)	Initiate Secure Software Update	Secure software update requested/pending (1) Secure software update complete (0)

3784 **Table 58 – Value Definition of the oic.sec.dpmttype Property (High-Byte)**

Value	Device Mode	Description
bx0000,0000 – bx1111,1111	<Reserved>	Reserved for later use

3785 The *provisioning operation mode* type is a 8-bit mask enumerating the various provisioning  
 3786 operation modes.

3787 "oic.sec.pomttype" is defined in Table 62.

3788

**Table 59 – Definition of the oic.sec.pomtype Property**

Type Name	Type URN	Description
Device Provisioning OperationMode	oic.sec.pomtype	Device provisioning operation mode is a 8-bit bitmask describing various provisioning operation modes

3789 Table 63 defines the values of "oic.sec.pomtype".

3790

**Table 60 – Value Definition of the oic.sec.pomtype Property**

Value	Operation Mode	Description
bx0000,0001 (1)	Server-directed utilizing multiple provisioning services	Provisioning related services are placed in different Devices. Hence, a provisioned Device should establish multiple DTLS sessions for each service. This condition exists when bit 0 is FALSE.
bx0000,0010 (2)	Server-directed utilizing a single provisioning service	All provisioning related services are in the same Device. Hence, instead of establishing multiple DTLS sessions with provisioning services, a provisioned Device establishes only one DTLS session with the Device. This condition exists when bit 0 is TRUE.
bx0000,0100 (4)	Client-directed provisioning	Device supports provisioning service control of this Device's provisioning operations. This condition exists when bit 1 is TRUE. When this bit is FALSE this Device controls provisioning steps.
bx0000,1000(8) – bx1000,0000(128)	<Reserved>	Reserved for later use
bx1111,11xx	<Reserved>	Reserved for later use

3791 **13.9 Certificate Signing Request Resource**

3792 The /oic/sec/csr Resource is used by a Device to provide its desired identity, public key to be  
 3793 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC  
 3794 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of  
 3795 /oic/sec/doxm Resource has a 1 in the 0x8 bit position), the Device shall have a /oic/sec/csr  
 3796 Resource.

3797 "oic.r.csr" Resource is defined in Table 64.

3798

**Table 61 – Definition of the oic.r.csr Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/csr	Certificate Signing Request	oic.r.csr	baseline	The CSR resource contains a Certificate Signing Request for the Device's public key.	Configuration

3799 Table 65 defines the Properties of "oic.r.csr".

3800

**Table 62 – Properties of the oic.r.csr Resource**

Property Title	Property Name	Value Type	Access Mode	Mandatory	Description
Certificate Signing Request	csr	String	R	Yes	Contains the signed CSR encoded according to the encoding Property
Encoding	encoding	String	R	Yes	A string specifying the encoding format of the data contained in the csr Property "oic.sec.encoding.pem" – Encoding for PEM-encoded certificate signing request "oic.sec.encoding.der" – Encoding for DER-encoded certificate signing request

3801 The Device chooses which public key to use, and may optionally generate a new key pair for this  
3802 purpose.

3803 In the CSR, the Common Name component of the Subject Name shall contain a string of the format  
3804 "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122. The  
3805 Common Name, and other components of the Subject Name, may contain other data. If the Device  
3806 chooses to include additional information in the Common Name component, it shall delimit it from  
3807 the UUID field by white space, a comma, or a semicolon.

3808 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to generate  
3809 a new key pair, the Device may begin generation of a key pair as a result of a RETRIEVE of this  
3810 resource. If the Device cannot immediately respond to the RETRIEVE request due to time required  
3811 to generate a key pair, the Device shall return an "operation pending" error. This indicates to the  
3812 Client that the Device is not yet ready to respond, but will be able at a later time. The Client should  
3813 retry the request after a short delay.

### 3814 13.10 Roles Resource

3815 The roles Resource maintains roles that have been asserted with role certificates, as described in  
3816 Clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role  
3817 certificate. Servers shall only grant access to the roles information associated with the public key  
3818 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.  
3819 See 10.4.2 for how role certificates are validated.

3820 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS session  
3821 with a Client, if is not already created. The roles Resource shall only expose a secured OCF  
3822 Endpoint in the /oic/res response. A Server shall retain the roles Resource at least as long as the  
3823 (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least until the  
3824 certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of clause  
3825 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A Server  
3826 should regularly inspect the contents of the roles resource and purge contents based on a policy it  
3827 determines based on its resource constraints. For example, expired certificates, and certificates  
3828 from Clients that have not been heard from for some arbitrary period of time could be candidates  
3829 for purging.

3830 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In  
3831 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall behave  
3832 as follows. Unlisted operations are implementation specific and not reliable.

3833 1) A RETRIEVE request shall return all previously asserted roles associated with the currently  
3834 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query  
3835 parameter is not supported; all previously asserted roles associated with the currently  
3836 connected and authenticated Client's identity are returned.

- 3837 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties  
 3838 included in the array as follows:
- 3839 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the  
 3840 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid" value.
- 3841 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,  
 3842 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed  
 3843 response and a duplicate entry shall not be added to the array.
- 3844 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored  
 3845 by the Server. The Server shall assign a unique "credid" value for every entry of the "roles"  
 3846 array.
- 3847 3) A DELETE request without a "credid" query parameter shall remove all entries from the  
 3848 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
 3849 Client's identity.
- 3850 4) A DELETE request with a "credid" query parameter shall remove only the entries of the  
 3851 /oic/sec/roles resource array corresponding to the currently connected and authenticated  
 3852 Client's identity and where the corresponding "credid" matches the entry.

3853 NOTE The oic.r.roles Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in  
 3854 ISO/IEC 30118-1:2018.

3855 "oic.r.roles" Resource is defined in Table 66.

3856 **Table 63 – Definition of the oic.r.roles Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/roles	Roles	oic.r.roles	baseline	Resource containing roles that have previously been asserted to this Server	Security

3857 Table 67 defines the Properties of "oic.r.roles".

3858 **Table 64 – Properties of the oic.r.roles Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Roles	roles	oic.sec.cred	array	RW	Yes	List of roles previously asserted to this Server

3859 Because oic.r.roles shares the oic.sec.cred schema with oic.r.cred, "subjectuud" is a required Property. However,  
 3860 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property  
 3861 is contained in an UPDATE request to the /oic/sec/roles Resource.

3862 **13.11 Account Resource**

3863 The Account Resource specifies the Properties based on IETF RFC 6749 Access Token based  
 3864 account creation. The mechanism to obtain credentials is described in Clause 7.5. The Account  
 3865 Resource is used for Device Registration. The Account Resource is instantiated on the OCF Cloud  
 3866 as "oic/sec/account" SVR and is used by cloud-enabled Devices to register with the OCF Cloud. It  
 3867 should be only accessible on a secure channel; non-secure channel should not be able access this  
 3868 Resource.

3869 During the Device Registration process, an OCF Cloud can provide a distinct URI of another OCF  
 3870 Cloud ("redirected-to" OCF Cloud). Both initial and redirected-to OCF Clouds are expected to  
 3871 belong to the same Vendor; they are assumed to have the same UUID and are assumed to have  
 3872 an out-of-band communication mechanism established. Device does not have to perform the  
 3873 Device Registration on the redirected-to OCF Cloud and the OCF Cloud may ignore such attempts.

3874 Redirected-to OCF Cloud is expected to accept the Access Token, provided to the Device by the  
3875 initial OCF Cloud.

3876 The "di", "uid", "refreshToken" and "accessToken" Properties of the Account Resource should be  
3877 securely stored as described in Clause 15.

3878 The RETRIEVE operation on OCF Cloud's "/oic/sec/account" Resource is not allowed and the OCF  
3879 Cloud is expected to reject all attempts to perform such operation.

3880 The UPDATE operation on the OCF Cloud's "/oic/sec/account" Resource behaves as follows:

3881 – A Device intending to register with the OCF Cloud shall send UPDATE with following Properties  
3882 "di" ("di" Property Value of "/oic/d" Resource), and "accessToken" as configured by the Mediator  
3883 ("at" Property Value of oic.r.coapcloudconf Resource). The OCF Cloud verifies it is the same  
3884 "accessToken" which was assigned to the Mediator for the corresponding "di" Property Value.  
3885 The "accessToken" is the permission for the Device to access the OCF Cloud. If the "apn" was  
3886 included when the Mediator UPDATED the "oic.r.coapcloudconf" Resource, the Device shall  
3887 also include "authprovider" Property when registering with the OCF Cloud. If no "apn" is  
3888 specified, then the "authprovider" Property shall not be included in the UPDATE request.

3889 – OCF Cloud returns "accessToken", "uid", "refreshToken", "expiresin" It may also return  
3890 "redirecturi". Received "accessToken" is to be treated by Device as an Access Token with  
3891 "Bearer" token type as defined in IETF RFC 6750. This "accessToken" shall be used for the  
3892 following Account Session start using "oic/sec/session" SVR. Received "refreshToken" is to be  
3893 treated by Device as a Refresh Token as defined in IETF RFC 6749. The Device stores the  
3894 OCF Cloud's Response values. If "redirecturi" is received, Device shall use received value as  
3895 a new OCF Cloud URI instead of "cis" Property Value of "oic.r.coapcloudconf" Resource for  
3896 further connections.

3897 The DELETE operation on the OCF Cloud's "/oic/sec/account" Resource should behave as follows:

3898 – To deregister with the OCF Cloud, a DELETE operation shall be sent with the "accessToken"  
3899 and either "uid", or "di" to be deregistered with the OCF Cloud. On DELETE with the OCF Cloud,  
3900 the Device should also delete values internally stored. Once deregister with an OCF Cloud,  
3901 Device can connect to any other OCF Cloud. Device deregistered need to go through the steps  
3902 in 7.5 again to be registered with the OCF Cloud.

3903 " oic.r.account " Resource is defined in Table 68.

3904 **Table 65 – Definition of the oic.r.account Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/account	Account	oic.r.account	oic.if.baseline	Resource used for a device to add itself under a given credential	N/A

3905 Table 69 defines the Properties of "oic.r.account ".

**Table 66 – Properties of the oic.r.account Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Device ID	di	string	uuid	W	Yes	Unique Device identifier
Auth Provider	authprovider	string	N/A	W	No	The name of Authorization Provider through which Access Token was obtained.
Access-Token	accesstoken	string	Non-empty string	RW	Yes	Access-Token used for communication with OCF Cloud after account creation
Refresh Token	refreshtoken	string	Non-empty string	R	Yes	Refresh token can be used to refresh the Access Token before getting expired
Token Expiration	expiresin	integer	-	R	Yes	Access-Token life time in seconds (-1 if permanent)
User ID	uid	string	uuid	R	Yes	Unique OCF Cloud User identifier
Redirect URI	redirecturi	string	-	R	No	Using this URI, the Client needs to reconnect to a redirected OCF Cloud. If provided, this value shall be used by the Device instead of Mediator-provided URI during the Device Registration.

**3907 13.12 Account Session resource**

3908 The "/oic/sec/session" Resource hosted on the OCF Cloud is used for creating connections with  
 3909 the OCF Cloud subsequent to Device registration though "/oic/sec/account" Resource. The  
 3910 "/oic/sec/session" Resource requires the device ID, User ID and Access Token which are stored  
 3911 securely on the Device.

3912 The /oic/sec/session Resource is exposed by the OCF Cloud. It should be only accessible on a  
 3913 secure channel; non-secure channel cannot access this Resource.

3914 The RETRIEVE operation on OCF Cloud's "/oic/sec/session" Resource is not allowed and the OCF  
 3915 Cloud is expected to reject all attempts to perform such operation.

3916 The UPDATE operation is defined as follows for OCF Cloud's "/oic/sec/session" Resource:

- 3917 – The Device connecting to the OCF Cloud shall send an UPDATE request message to the OCF  
 3918 Cloud's /oic/sec/session Resource. The message shall include the "di" Property Value of /oic/d  
 3919 Resource and "uid", "login" Value ("true" to establish connection; "false" to disconnect) and  
 3920 "accesstoken" as returned by OCF Cloud during Device Registration. The OCF Cloud verifies  
 3921 it is the same Access Token which was returned to the Device during Device Registration  
 3922 process. If Device was attempting to establish the connection and provided values were verified  
 3923 as correct by the OCF Cloud, OCF Cloud sends a response with remaining lifetime of the  
 3924 associated Access Token ("expiresin" Property Value).

3925 "oic.r.session" Resource is defined in Table 70.

3926

**Table 67 – Definition of the oic.r.session Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/session	Account Session	oic.r.session	oic.if.baseline	Resource that enables a device to manage its session using login or logout	N/A

3927 Table 71 defines the Properties of "oic.r.session".

3928

**Table 68 – Properties of the oic.r.session Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
User ID	uid	string	uuid	W	Yes	User ID which provided by Device Registration process
Device ID	di	string	uuid	W	Yes	Unique device id registered for a Device
Access Token	accesstoken	string	A string of at least one character	W	Yes	Access-Token used to grant access right for the Device to login/sign-in
Login Status	login	boolean	N/A	W	Yes	Action for the request: true = login, false = logout
Token Expiration	expiresin	integer	N/A	R	Yes	Remaining Access-Token life time in seconds (-1 if permanent) This Property is only provided to Device during connection establishment (when "login" Property Value equals "true"), it's not available otherwise

3929 **13.13 Account Token Refresh Resource**

3930 The "/oic/sec/tokenrefresh" Resource is used by the Device for refreshing the Access Token.

3931 The "/oic/sec/tokenrefresh" Resource is hosted by the OCF Cloud. It should be only accessible on  
3932 a secure channel; non-secure channel cannot access this Resource.

3933 The Device should use "/oic/sec/tokenrefresh" to refresh the Access Token with the OCF Cloud,  
3934 when the time specified in "expiresin" is near.

3935 The RETRIEVE operation on OCF Cloud's "/oic/sec/ tokenrefresh" Resource is not allowed and the  
3936 OCF Cloud is expected to reject all attempts to perform such operation.

3937 The UPDATE operation is defined as follows for "/oic/sec/tokenrefresh" Resource

- 3938 – The Device attempting to refresh the Access Token shall send an UPDATE request message  
3939 to the OCF Cloud's /oic/sec/tokenrefresh Resource. The message shall include the "di" Property  
3940 Value of /oic/d Resource, "uid" and "refreshtoken", as returned by OCF Cloud.
- 3941 – OCF Cloud response is expected to include a "refreshtoken", new "accesstoken", and  
3942 "expiresin". Received "accesstoken" is to be treated by Device as an Access Token with "Bearer"  
3943 token type as defined in IETF RFC 6750. This Access Token is the permission for the Device  
3944 to access the OCF Cloud. Received "refreshtoken" is to be treated by Device as a Refresh  
3945 Token as defined in IETF RFC 6749. Received "refreshtoken" may be the new Refresh Token  
3946 or the same one as provided by the Device in the UPDATE request. In case when new distinct  
3947 "refreshtoken" is provided by the OCF Cloud, the Device shall discard the old value. The OCF  
3948 Cloud's response values "refreshtoken", "accesstoken" and "expiresin" are securely stored on  
3949 the Device.

3950 "/oic.r.tokenrefresh" Resource is defined in Table 72.

3951

**Table 69 – Definition of the oic.r.tokenrefresh Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/tokenrefresh	Token Refresh	oic.r.tokenrefresh	oic.if.baseline	Resource to manage the access-token using refresh token	N/A

3952 Table 73 defines the Properties of "oic.r.tokenrefresh".

3953

**Table 70 – Properties of the oic.r.tokenrefresh Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
User ID	uid	string	uuid	W	Yes	User ID which provided by Sign-up process
Device ID	di	string	uuid	W	Yes	Unique device id registered for an OCF Cloud User account
Refresh Token	refresh token	string	A string of at least one character	RW	Yes	Refresh token received by account management or during token refresh procedure
Access Token	access token	string	A string of at least one character	R	Yes	Granted Access-Token
Token Expiration	expires in	integer	-	R	Yes	Access-Token life time in seconds (-1 if permanent)

3954 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3955 The SVRs expose the security-related Properties of the Device.

3956 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to unauthenticated  
3957 (anonymous) Clients could create privacy or security concerns.3958 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for  
3959 the oic.r.doxm Resource to anonymous requesters, so that the Device can be discovered and  
3960 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the oic.r.doxm  
3961 Resource to anonymous requesters, to preserve privacy.3962 **13.15 SVRs, Discoverability and OCF Endpoints**3963 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy Parameter  
3964 clause 7.8.2.1.2).3965 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS) (reference  
3966 ISO/IEC 30118-1:2018, clause 10).3967 The /oic/sec/doxm Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM  
3968 (reference ISO/IEC 30118-1:2018, clause 10).3969 **13.16 Additional Privacy Consideration for Core and SVRs Resources**3970 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**3971 Unique identifiers are a privacy consideration due to their potential for being used as a tracking  
3972 mechanism. These include the following Resources and Properties:

3973 – /oic/d Resource containing the 'di' and 'piid' Properties.  
3974 – /oic/p Resource containing the 'pi' Property.  
3975 – /oic/sec/doxm Resource containing the 'deviceuuid' Property.  
3976 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous  
3977 requestors. This implies any Client Device, including those with malicious intent, are able to reliably  
3978 obtain identifiers useful for building a log of activity correlated with a specific Platform and Device.

3979 There are two strategies for privacy protection of Devices:

3980 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers  
3981 2) Limit identifier persistence to make it impractical for tracking use.  
3982 Both techniques can be used effectively together to limit exposure to privacy attacks.

3983 1) A Platform / Device manufacturer should specify a default ACL policy that restricts anonymous  
3984 requestors from accessing unique identifiers. An OCF Security Domain owner should modify  
3985 the ACL policy to grant access to authenticated Devices who, presumably, do not present a  
3986 privacy threat.

3987 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the  
3988 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and  
3989 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated  
3990 identifiers shall be:

3991 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers  
3992 b) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
3993 resistant

3994 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier used  
3995 to begin the onboarding process. However, attackers could obtain the value too and use it for  
3996 Device tracking throughout the Device's lifetime.

3997 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the  
3998 deviceuuid Property of the /oic/sec/doxm Resource to unauthenticated /oic/res and /oic/sec/doxm  
3999 Resource RETRIEVE requests when the devowneruuid Property of /oic/sec/doxm Resource is the  
4000 nil-UUID. The Server shall expose a new temporary non-repeated deviceuuid Property of the  
4001 /oic/sec/doxm Resource when the device state transitions to RESET. This ensures the deviceuuid  
4002 Property of the /oic/sec/doxm cannot be used to track across multiple owners.

4003 The devowneruuid Property of /oic/sec/doxm Resource is initialized to the nil-UUID upon entering  
4004 RESET; which is retained until being set to a non-nil-UUID value during RFOTM device state. The  
4005 device shall supply a temporary, non-repeated deviceuuid Property of /oic/sec/doxm Resource to  
4006 RETRIEVE requests on /oic/sec/doxm and /oic/res Resources while devowneruuid Property of  
4007 /oic/sec/doxm Resource is the nil-UUID. During the OTM process the DOTS shall UPDATE  
4008 devowneruuid Property of the /oic/sec/doxm Resource to a non-nil UUID value which is the trigger  
4009 for the Device to expose its persistent or semi-persistent device identifier. Therefore the Device  
4010 shall supply deviceuuid Property of /oic/sec/doxm Resource in response to RETRIEVE requests  
4011 while the devowneruuid Property of the /oic/sec/doxm Resource is a non nil-UUID value.

4012 The DOTS or AMS may also provision an ACL policy that restricts access to the /oic/sec/doxm  
4013 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent  
4014 device identifier via the deviceuuid Property value of the /oic/sec/doxm Resource.

4015 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent  
4016 or semi-persistent identifier using the /oic/sec/cred Resource to first establish an authenticated  
4017 connection. This is achieved by first provisioning a /oic/sec/cred Resource entry that contains the  
4018 Server's deviceuuid Property value of the /oic/sec/doxm Resource.

4019 The di Property in the /oic/d Resource shall mirror that of the deviceuuid Property of the  
 4020 /oic/sec/doxm Resource. The DOTS should provision an ACL policy that restricts access to the  
 4021 /oic/d resource such that only authenticated Clients are able to obtain the di Property of /oic/d  
 4022 Resource. See Clause 13.1 for deviceuuid Property lifecycle requirements.

4023 Servers should expose a temporary, non-repeated, piid Property of /oic/p Resource Value upon  
 4024 entering RESET Device state. Servers shall expose a persistent value via the piid Property of /oic/p  
 4025 Property when the DOTS sets devowneruuid Property to a non-nil-UUID value. An ACL policy on  
 4026 the /oic/d Resource should protect the piid Property of /oic/p Resource from being disclosed to  
 4027 unauthenticated requestors.

4028 Servers shall expose a temporary, non-repeated, pi Property value upon entering RESET Device  
 4029 state. Servers shall expose a persistent or semi-persistent platform identifier value via the pi  
 4030 Property of the /oic/p Resource when onboarding sets devowneruuid Property to a non-nil-UUID  
 4031 value. An ACL policy on the /oic/p Resource should protect the pi Property from being disclosed to  
 4032 unauthenticated requestors.

4033 Table 74 depicts Core Resource Properties Access Modes given various Device States.

**Table 71 – Core Resource Properties Access Modes given various Device States**

Resource Type	Property title	Property name	Value type	Access Mode		Behaviour
oic.wk.p	Platform ID	pi	oic.types-schema.uuid	All States	R	Server shall construct a temporary random UUID (The temporary value shall not overwrite the persistent pi internally). Server sets to its persistent value after secure Owner Transfer session is established.
oic.wk.d	Protocol Independent Identifier	piid	oic.types-schema.uuid	All States	R	Server should construct a temporary random UUID when entering RESET state.
oic.wk.d	Device Identifier	di	oic.types-schema.uuid	All states	R	/d di shall mirror the value contained in /doxm deviceuuid in all device states.

4035 Four identifiers are thought to be privacy sensitive:

- 4036 – /oic/d Resource containing the 'di' and 'piid' Properties.
- 4037 – /oic/p Resource containing the 'pi' Property.
- 4038 – /oic/sec/doxm Resource containing the 'deviceuuid' Property.

4039 There are three strategies for privacy protection of Devices:

- 4040 1) Apply access control to restrict read access to Resources containing unique identifiers. This  
 4041 ensures privacy sensitive identifiers do not leave the Device.
- 4042 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy sensitive  
 4043 identifiers are less effective for tracking and correlation.
- 4044 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can  
 4045 do so.

4046 These techniques can be used to limit exposure to privacy attacks. For example:

4047 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent  
4048 identifiers can be created.

4049 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to  
4050 facilitate onboarding.

4051 – Persistent and semi-persistent identifiers can be encrypted before sending them to another  
4052 Device.

4053 A temporary, non-repeated identifier shall be:

4054 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers

4055 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
4056 resistant

4057 NOTE This requirement is met through a vendor attestation certification mechanism.

### 4058 13.16.2 Privacy Protecting the Device Identifiers

4059 The "di" Property Value of the /oic/d Resource shall mirror that of the "deviceuuid" Property of the  
4060 /oic/sec/doxm Resource. The Device should use a new, temporary non-repeated identifier in place  
4061 of the "deviceuuid" Property Value of /oic/sec/doxm Resource upon entering the RESET Device  
4062 state. This value should be exposed while the "devowneruuid" Property has a nil UUID value. The  
4063 Device should expose its persistent (or semi-persistent) "deviceuuid" Property value of the  
4064 /oic/sec/doxm Resource after the DOTS sets the "devowneruuid" Property to a non-nil-UUID value.  
4065 The temporary identifier should not change more frequently than once per Device state transition  
4066 to RESET.

4067 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

4068 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"  
4069 Property values:

4070 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di") Property  
4071 value only if responding to an authenticated requestor and the "deviceuuid" (or "di") value  
4072 is confidentiality protected .

4073 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if  
4074 responding to an unauthenticated requestor.

4075 – The AMS should provision an ACL policy on the /oic/sec/doxm and /oic/d resources to further  
4076 protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

4077 See 13.2 for deviceuuid Property lifecycle requirements.

4078 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary  
4079 discovery requests. This is achieved by provisioning a /oic/sec/cred Resource entry that contains the Server's deviceuuid  
4080 Property value. The Client establishes a secure connection to the Server straight away.

### 4081 13.16.3 Privacy Protecting the Protocol Independent Device Identifier

4082 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property  
4083 Value of /oic/d Resource upon entering the RESET Device state. If a temporary, non-repeated  
4084 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID  
4085 value. The Device should use its persistent "piid" Property value after the DOTS sets the  
4086 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change more  
4087 frequently than once per Device state transition to RESET.

4088 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

4089 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:

4090 – The Device should include its persistent "piid" Property value only if responding to an  
4091 authenticated requestor and the "piid" value is confidentiality protected.

- 4092 – The Device should include a temporary non-repeated "piid" Property value if responding to  
4093 an unauthenticated requestor.
- 4094 – The AMS should provision an ACL policy on the /oic/d Resource to further protect the piid  
4095 Property of /oic/p Resource from being disclosed unnecessarily.

4096 **13.16.4 Privacy Protecting the Platform Identifier**

4097 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property Value  
4098 of the /oic/p Resource upon entering the RESET Device state. This value should be exposed while  
4099 the "devowneruuid" Property has a nil UUID value. The Device should use its persistent (or semi-  
4100 persistent) "pi" Property value after the DOTS sets the "devowneruuid" Property to a non-nil-UUID  
4101 value. The temporary identifier should not change more frequently than once per Device state  
4102 transition to RESET.

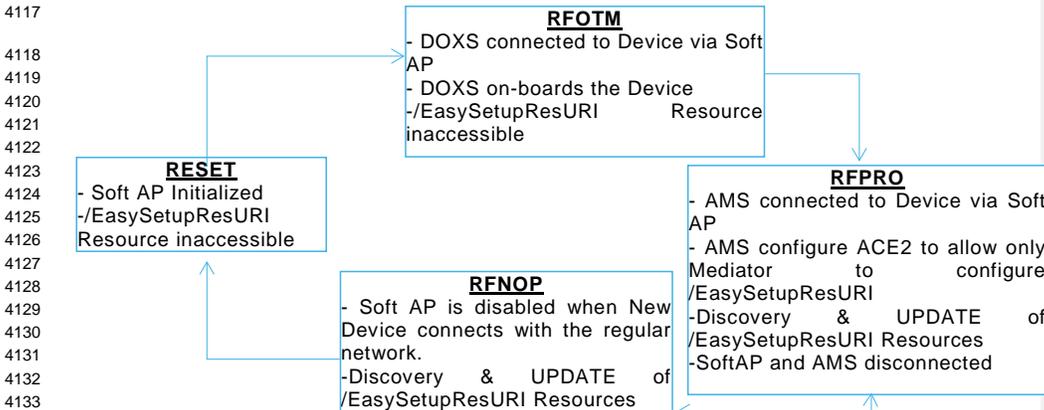
4103 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4104 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:  
4105 – The Device should include its persistent (or semi-persistent) "pi" Property value only if  
4106 responding to an authenticated requestor and the "pi" value is confidentiality protected.
- 4107 – The Device should include a temporary non-repeated "pi" Property value if responding to an  
4108 unauthenticated requestor.
- 4109 – The AMS should provision an ACL policy on the /oic/p Resource to protect the pi Property from  
4110 being disclosed unnecessarily.

4111 **13.17 Easy Setup Resource Device State**

4112 This clause only applies to New Device that uses Easy Setup for Ownership Transfer as defined in  
4113 ISO/IEC 30118-7:2018. Easy setup has no impact to New Devices that have a different way of  
4114 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup  
4115 Devices.

4116 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



4134 **Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states**

4135 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's  
4136 state.

4137 While it is reasonable for a user to expect that power cycling a New Device will turn on the Soft AP  
4138 for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it is a

4139 security risk to make this the default behavior of a device that remains unenrolled beyond a  
4140 reasonable period after first boot.

4141 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

4142 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes  
4143 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for Easy  
4144 Setup.

4145 – If a New Device tried and failed to complete Easy Setup Enrollment immediately following the  
4146 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically for  
4147 another 30 mins upon being power cycled, provided that the power cycle occurs within 3 hours  
4148 of first boot or the most recent factory reset. If the user has initiated the Easy Setup Soft AP  
4149 directly without a factory reset, it is not necessary to turn it back on if it was on immediately  
4150 prior to power cycle, because the user obviously knows how to initiate the process manually.

4151 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft  
4152 AP should not turn back on for Easy Setup until another factory reset occurs, or the user initiates  
4153 the Easy Setup Soft AP directly.

4154 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the New  
4155 Device to connect to the Enroller.

4156 – The Easy Setup Soft AP shall be disabled when the New Device successfully connects to the  
4157 Enroller.

4158 – Once a New Device has successfully connected to the Enroller, it shall not turn the Easy Setup  
4159 Soft AP back on for Easy Setup Enrollment again unless the Device is factory reset, or the user  
4160 initiates the Easy Setup Soft AP directly.

4161 – Just Works OTM shall not be enabled on Devices which support Easy Setup.

4162 – The Soft AP shall be secured (e.g. shall not expose an open AP).

4163 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase  
4164 shall be between 8 and 64 ASCII printable characters. The passphrase may be printed on  
4165 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.

4166 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the  
4167 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by an  
4168 attacker with knowledge of the Device type, model, manufacturer, or any other information  
4169 discoverable through Device's exposed interfaces.

4170 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the  
4171 /example/WiFiConfResURI Resource), for potential selection by the Mediator in connecting the  
4172 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller, for  
4173 use in connecting the Enrollee to the Enroller.

4174 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the  
4175 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.

4176 The /example/EasySetupResURI Resource should not be discoverable in RFOTM or SRESET state.  
4177 After Ownership Transfer process is completed with the DOTS, and the Device enters in RFPRO  
4178 Device state, the /example/EasySetupResURI may be Discoverable. The DOTS may be hosted on  
4179 the Mediator Device.

4180 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership  
4181 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used by  
4182 AMS for /oic/sec/acl2 Resource provisioning in RFPRO state. The CoAPS session authentication  
4183 and encryption is already defined in the Security spec.

4184 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following  
4185 Resources to be only configurable by the Mediator Device with permission to UPDATE or  
4186 RETRIEVE access:

4187 – /example/EasySetupResURI

4188 – /example/WifiConfResURI

4189 – /example/DevConfResURI

4190 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
4191 {  
4192     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
4193     "resources": [  
4194         { "href": "/example/EasySetupResURI" },  
4195         { "href": "/example/WiFiConfResURI" },  
4196         { "href": "/example/DevConfResURI" },  
4197     ],  
4198     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
4199 }
```

4200 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to  
4201 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

4202 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE  
4203 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

## 4204 **14 Security Hardening Guidelines/ Execution Environment Security**

### 4205 **14.1 Preamble**

4206 This is an informative clause. Many TGs in OCF have security considerations for their protocols  
4207 and environments. These security considerations are addressed through security mechanisms  
4208 specified in the security documents for OCF. However, effectiveness of these mechanisms depends  
4209 on security robustness of the underlying hardware and software Platform. This clause defines the  
4210 components required for execution environment security.

### 4211 **14.2 Execution Environment Elements**

#### 4212 **14.2.1 Execution Environment Elements General**

4213 Execution environment within a computing Device has many components. To perform security  
4214 functions in a robustness manner, each of these components has to be secured as a separate  
4215 dimension. For instance, an execution environment performing AES cannot be considered secure  
4216 if the input path entering keys into the execution engine is not secured, even though the partitions  
4217 of the CPU, performing the AES encryption, operate in isolation from other processes. Different  
4218 dimensions referred to as elements of the execution environment are listed below. To qualify as a  
4219 secure execution environment (SEE), the corresponding SEE element must qualify as secure.

- 4220 – (Secure) Storage
- 4221 – (Secure) Execution engine
- 4222 – (Trusted) Input/output paths
- 4223 – (Secure) Time Source/clock
- 4224 – (Random) number generator
- 4225 – (Approved) cryptographic algorithms
- 4226 – Hardware Tamper (protection)

4227 NOTE Software security practices (such as those covered by OWASP) are outside scope of this document, as  
4228 development of secure code is a practice to be followed by the open source development community. This document will  
4229 however address the underlying Platform assistance required for executing software. Examples are secure boot and  
4230 secure software upgrade.

4231 Each of the elements above are described in the clauses 14.2.2, 14.2.3, 14.2.4, 14.2.5, 14.2.6,  
4232 14.2.7.

#### 4233 **14.2.2 Secure Storage**

##### 4234 **14.2.2.1 Secure Storage General**

4235 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive  
4236 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,  
4237 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive  
4238 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both its  
4239 integrity and confidentiality be maintained.

4240 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive  
4241 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either  
4242 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication and  
4243 encryption, it must maintain its integrity against intentional or accidental alteration.

4244 A partial list of Sensitive Data is outlined in Table 75:

**Table 72 – Examples of Sensitive Data**

<b>Data</b>	<b>Integrity protection</b>	<b>Confidentiality protection</b>
Owner PSK (Symmetric Keys)	Yes	Yes
Service provisioning keys	Yes	Yes
Asymmetric Private Keys	Yes	Yes
Certificate Data and Signed Hashes	Yes	Not required
Public Keys	Yes	Not required
Access credentials (e.g. SSID, passwords, etc.)	Yes	Yes
ECDH/ECDH Dynamic Shared Key	Yes	Yes
Root CA Public Keys	Yes	Not required
Device and Platform IDs	Yes	Not required
Easy Setup Resources	Yes	Yes
OCF Cloud URL	Yes	Not required
OCF Cloud Identity	Yes	Not required
Access Token	Yes	Yes

4246 Exact method of protection for secure storage is implementation specific, but typically combinations  
4247 of hardware and software methods are used.

#### 4248 **14.2.2.2 Hardware Secure Storage**

4249 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric  
4250 and asymmetric private keys, access credentials, and personal private data. Hardware secure  
4251 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes  
4252 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

4253 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides  
4254 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or electronic  
4255 attacks. It is not necessary to prevent the attacks themselves, but an attempted attack should not  
4256 result in an unauthorized entity successfully retrieving Sensitive Data.

4257 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data  
4258 from attacks that include but are not limited to:

- 4259 1) Physical decapping of chip packages to optically read NVRAM contents
- 4260 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 4261 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit patterns  
4262 of Critical Sensitive Data
- 4263 4) Use of malicious software or firmware to read memory contents at rest or in transit within a  
4264 microcontroller
- 4265 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

#### 4266 **14.2.2.3 Software Storage**

4267 It is generally NOT recommended to rely solely on software and unsecured memory to store  
4268 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and encryption  
4269 keys should be housed in hardware secure storage whenever possible.

4270 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable  
4271 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

#### 4272 **14.2.2.4 Additional Security Guidelines and Best Practices**

4273 Some general practices that can help ensure that Sensitive Data is not compromised by various  
4274 forms of security attacks:

- 4275 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG  
4276 used for authentication challenges can substantially degrade security strength. For this reason,  
4277 it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source be used  
4278 for all authentication challenges.
- 4279 2) Secure download and boot – To prevent the loading and execution of malicious software, where  
4280 it is practical, it is recommended that Secure Download and Secure Boot methods that  
4281 authenticate a binary's source as well as its contents be used.
- 4282 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered  
4283 unsecure and shall not be used for any security-related function:
  - 4284 a) SHA-1
  - 4285 b) MD5
  - 4286 c) RC4
  - 4287 d) RSA 1024
- 4288 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is  
4289 stored in Secure Storage, any use of that data that requires its transmission out of that Secure  
4290 Storage should be encrypted to prevent eavesdropping by malicious software within an  
4291 MCU/MPU.
- 4292 5) It is recommended to avoid using wildcard in Subject Id ("\*"), when setting up oic.r.cred  
4293 Resource entries, since this opens up an identity spoofing opportunity.
- 4294 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device  
4295 meets security requirements for its intended uses. As an example, IoTivity is a reference  
4296 implementation intended to be used as a basis for a product, but IoTivity has not undergone  
4297 3rd party security review, penetration testing, etc. Any Device based on IoTivity should undergo  
4298 appropriate penetration testing and security review prior to sale or deployment.
- 4299 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to  
4300 consumers. Changes should be made to a public and accessible website. Expectations should  
4301 be clear as to what will be supported and for how long the Device vendor expects to support  
4302 security updates to the software, operating system, drivers, networking, firmware and hardware  
4303 of the device.
- 4304 8) Device vendor has not implemented test or debug interfaces on the Device which are operable  
4305 or which can be enabled which might present an attack vector on the Device which circumvents  
4306 the interface-level security or access policies of the Device.
- 4307 9) Device vendor understands that if an application running on the Device has access to  
4308 cryptographic elements such as the private keys or Ownership Credential, then those elements  
4309 have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or a Device  
4310 with access to the Internet beyond the local network, the execution of critical functions should  
4311 take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4312 10) Any PINs or fixed passphrases used for onboarding, WiFi Easy Setup, SoftAP management or  
4313 access, or other security-critical function, should be sufficiently unique (do not duplicate  
4314 passphrases. The creation of these passphrases or PINS should not be algorithmically  
4315 deterministic nor should they use insufficient entropy in their creation.
- 4316 11) Ensure that there are no remaining "VENDOR\_TODO" items in the source code.

4317 12) If the implementation of this document uses the "Just Works" onboarding method, understand  
4318 that there is a man-in-the-middle vulnerability during the onboarding process where a malicious  
4319 party could intercept messages between the device being onboarded and the OBT and could  
4320 persist, acting as an intermediary with access to message traffic, during the lifetime of that  
4321 onboarded device. The recommended best practice would be to use an alternate ownership  
4322 transfer method (OTM) instead of "Just Works".

4323 13) It is recommended that at least one static and dynamic analysis tool<sup>1</sup> be applied to any  
4324 proposed major production release of the software before its release, and any vulnerabilities  
4325 resolved.

4326 14) To avoid a malicious device being able to covertly join an OCF Security Domain, implementers  
4327 of any onboarding tool may eliminate completely autonomous sequences where a device is  
4328 brought into the OCF Security Domain without any authorization by the owner. Consider either  
4329 including a confirmation with the OCF Security Domain owner/operator (e.g. "Do you want to  
4330 add 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'? Yes/No/Cancel?") or a  
4331 confirmation with a security policy (e.g. an enterprise policy where the OCF Security Domain  
4332 admin can bulk-onboard devices).

#### 4333 14.2.3 Secure execution engine

4334 Execution engine is the part of computing Platform that processes security functions, such as  
4335 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine requires  
4336 the following

4337 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This  
4338 includes isolation of CPU caches, and all of execution elements that needed to be considered  
4339 as part of trusted (crypto) boundary.

4340 – Isolation of data paths into and out of execution engine. For instance both unencrypted but  
4341 sensitive data prior to encryption or after decryption, or cryptographic keys used for  
4342 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

#### 4343 14.2.4 Trusted input/output paths

4344 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be protected.  
4345 This includes paths into and out secure execution engine and secure memory.

4346 Path protection can be both hardware based (e.g. use of a privileged bus) or software based (using  
4347 encryption over an untrusted bus).

#### 4348 14.2.5 Secure clock

4349 Many security functions depend on time-sensitive credentials. Examples are time stamped  
4350 Kerberos tickets, OAUTH tokens, X.509 certificates, OSCP response, software upgrades, etc. Lack  
4351 of secure source of clock can mean an attacker can modify the system clock and fool the validation  
4352 mechanism. Thus an SEE needs to provide a secure source of time that is protected from tampering.  
4353 Trustworthiness from security robustness standpoint is not the same as accuracy. Protocols such  
4354 as NTP can provide rather accurate time sources from the network, but are not immune to attacks.  
4355 A secure time source on the other hand can be off by seconds or minutes depending on the time-  
4356 sensitivity of the corresponding security mechanism. Secure time source can be external as long  
4357 as it is signed by a trusted source and the signature validation in the local Device is a trusted  
4358 process (e.g. backed by secure boot).

#### 4359 14.2.6 Approved algorithms

4360 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and  
4361 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by  
4362 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only  
4363 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic

<sup>1</sup> A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4364 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are  
4365 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms (even  
4366 if they deemed stronger by some parties) must be considered non-approved.

4367 The set of algorithms to be considered for approval are algorithms for

- 4368 – Hash functions
- 4369 – Signature algorithms
- 4370 – Encryption algorithms
- 4371 – Key exchange algorithms
- 4372 – Pseudo Random functions (PRF) used for key derivation

4373 This list will be included in this or a separate security robustness rules document and must be  
4374 followed for all security specifications within OCF.

#### 4375 **14.2.7 Hardware tamper protection**

4376 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not  
4377 requirements) regarding tamper protection for cryptographic module

- 4378 – Production-grade (lowest level): this means components that include conformal sealing coating  
4379 applied over the module's circuitry to protect against environmental or other physical damage.  
4380 This does not however require zeroization of secret material during physical maintenance. This  
4381 definition is borrowed from FIPS 140-2 security level 1.
- 4382 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,  
4383 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from FIPS  
4384 140-2 security level 2.
- 4385 – Tamper resistance (highest level), this means there is a response to physical tempering that  
4386 typically includes zeroization of sensitive material on the module. This definition is borrowed  
4387 from FIPS 140-2 security level 3.

4388 It is difficult of specify quantitative certification test cases for accreditation of these levels. Content  
4389 protection regimes usually talk about different tools (widely available, specialized and professional  
4390 tools) used to circumvent the hardware protections put in place by manufacturing. If needed, OCF  
4391 can follow that model, if and when OCF engage in distributing sensitive key material (e.g. PKI) to  
4392 its members.

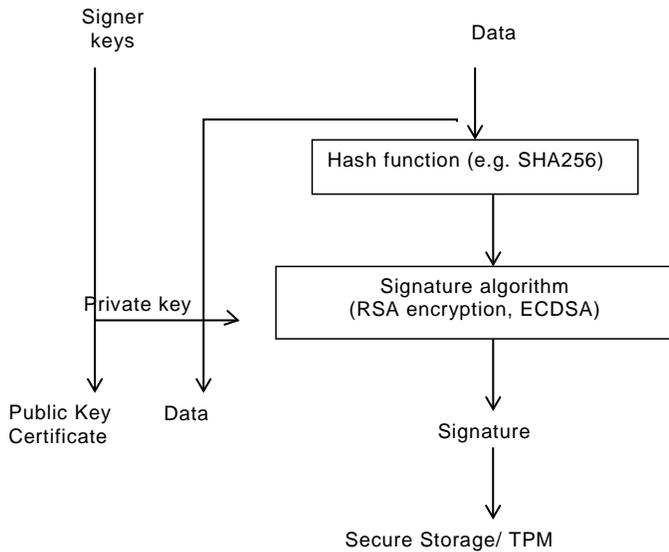
### 4393 **14.3 Secure Boot**

#### 4394 **14.3.1 Concept of software module authentication**

4395 In order to ensure that all components of a Device are operating properly and have not been  
4396 tampered with, it is best to ensure that the Device is booted properly. There may be multiple stages  
4397 of boot. The end result is an application running on top an operating system that takes advantage  
4398 of memory, CPU and peripherals through drivers.

4399 The general concept is the each software module is invoked only after cryptographic integrity  
4400 verification is complete. The integrity verification relies on the software module having been hashed  
4401 (e.g. SHA\_1, SHA\_256) and then signed with a cryptographic signature algorithm with (e.g. RSA),  
4402 with a key that only a signing authority has access to.

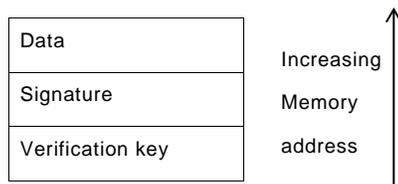
4403 Figure 40 depicts software module authentication.



**Figure 40 – Software Module Authentication**

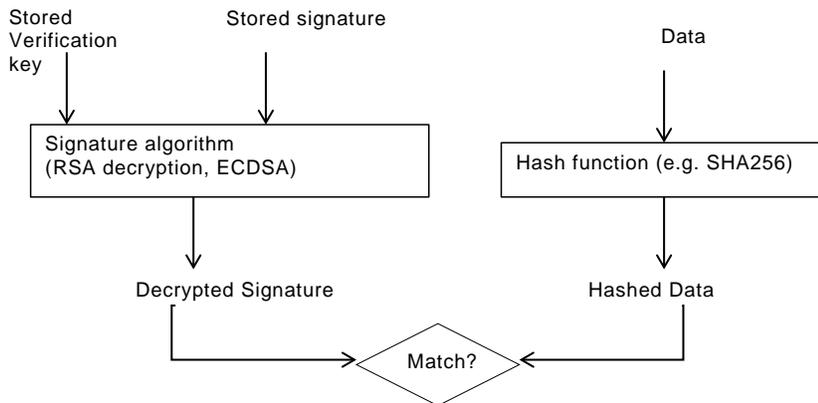
4404  
 4405 After the data is signed with the signer’s signing key (a private key), the verification key (the public  
 4406 key corresponding to the private signing key) is provided for later verification. For lower level  
 4407 software modules, such as bootloaders, the signatures and verification keys are inserted inside  
 4408 tamper proof memory, such as one-time programmable memory or TPM. For higher level software  
 4409 modules, such as application software, the signing is typically performed according to the PKCS#7  
 4410 format IETF RFC 2315, where the signedData format includes both indications for signature  
 4411 algorithm, hash algorithm as well as the signature verification key (or certificate). Secure boot does  
 4412 not require use of PKCS#7 format.

4413 Figure 41 depicts verification software module.



**Figure 41 – Verification Software Module**

4414  
 4415 As shown in Figure 42. the verification module first decrypts the signature with the verification key  
 4416 (public key of the signer). The verification module also calculates a hash of the data and then  
 4417 compares the decrypted signature (the original) with the hash of data (actual) and if the two values  
 4418 match, the software module is authentic.



4419 **Figure 42 – Software Module Authenticity**

4420 **14.3.2 Secure Boot process**

4421 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/  
 4422 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to  
 4423 find out where the boot code is and then run the boot code (second-stage boot loader). The second  
 4424 stage bootloader is typically the process that loads the operating system (Kernel) and transfers the  
 4425 execution to the where the Kernel code is. Once the Kernel starts, it may load external Kernel  
 4426 modules and drivers.

4427 When performing a secure boot, it is required that the integrity of each boot loader is verified before  
 4428 executing the boot loader stage. As mentioned, while the signature and verification key for the  
 4429 lowest level bootloader is typically stored in tamper-proof memory, the signature and verification  
 4430 key for higher levels should be embedded (but attached in an easily accessible manner) in the data  
 4431 structures software.

4432 **14.3.3 Robustness Requirements**

4433 **14.3.3.1 Robustness General**

4434 To qualify as high robustness secure boot process, the signature and hash algorithms shall be one  
 4435 of the approved algorithms, the signature values and the keys used for verification shall be stored  
 4436 in secure storage and the algorithms shall run inside a secure execution environment and the keys  
 4437 shall be provided the SEE over trusted path.

4438 **14.3.3.2 Next steps**

4439 Develop a list of approved algorithms and data formats

4440 **14.4 Attestation**

4441 **14.5 Software Update**

4442 **14.5.1 Overview:**

4443 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;  
 4444 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and  
 4445 end-of-life stages for the Device remain outstanding. It is possible for the Device to require update

4446 during any of these stages, although the most likely times are during onboarding, regular operation  
4447 and maintenance. The aspects of the software include, but are not limited to, firmware, operating  
4448 system, networking stack, application code, drivers, etc.

#### 4449 **14.5.2 Recognition of Current Differences**

4450 Different manufacturers approach software update utilizing a collection of tools and strategies:  
4451 over-the-air or wired USB connections, full or partial replacement of existing software, signed and  
4452 verified code, attestation of the delivery package, verification of the source of the code, package  
4453 structures for the software, etc.

4454 It is recommended that manufacturers review their processes and technologies for compliance with  
4455 industry best-practices that a thorough security review of these takes place and that periodic review  
4456 continue after the initial architecture has been established.

4457 This document applies to software updates as recommended to be implemented by Devices; it  
4458 does not have any bearing on the above-mentioned alternative proprietary software update  
4459 mechanisms.

#### 4460 **14.5.3 Software Version Validation**

4461 Setting the Initiate Software Version Validation bit in the `/oic/sec/pstat.tm` Property (see Table 57  
4462 defines the Properties of "oic.r.pstat").

4463 Table 57 of 13.8) indicates a request to initiate the software version validation process, the process  
4464 whereby the Device validates the software (including firmware, operating system, Device drivers,  
4465 networking stack, etc.) against a trusted source to see if, at the conclusion of the check, the  
4466 software update process will need to be triggered (see clause 14.5.4). When the Initiate Software  
4467 Version Validation bit of `/oic/sec/pstat.tm` is set to 1 (TRUE) by a sufficiently privileged Client, the  
4468 Device sets the `/oic/sec/pstat.cm` Initiate Software Version Validation bit to 0 and initiates a  
4469 software version check. Once the Device has determined if an update is available, it sets the Initiate  
4470 Software Version Validation bit in the `/oic/sec/pstat.cm` Property to 1 (TRUE) if an update is  
4471 available or 0 (FALSE) if no update is available. To signal completion of the Software Version  
4472 Validation process, the Device sets the Initiate Software Version Validation bit in the  
4473 `/oic/sec/pstat.tm` Property back to 0 (FALSE). If the Initiate Software Version Validation bit of  
4474 `/oic/sec/pstat.tm` is set to 0 (FALSE) by a Client, it has no effect on the validation process.

#### 4475 **14.5.4 Software Update**

4476 Setting the Initiate Secure Software Update bit in the `/oic/sec/pstat.tm` Property (see Table 57  
4477 defines the Properties of "oic.r.pstat").

4478 Table 57 of 13.8) indicates a request to initiate the software update process. When the Initiate  
4479 Secure Software Update bit of `/oic/sec/pstat.tm` is set to 1 (TRUE) by a sufficiently privileged Client,  
4480 the Device sets the `/oic/sec/pstat.cm` Initiate Software Version Validation bit to 0 and initiates a  
4481 software update process. Once the Device has completed the software update process, it sets the  
4482 Initiate Secure Software Update bit in the `/oic/sec/pstat.cm` Property to 1 (TRUE) if/when the  
4483 software was successfully updated or 0 (FALSE) if no update was performed. To signal completion  
4484 of the Secure Software Update process, the Device sets the Initiate Secure Software Update bit in  
4485 the `/oic/sec/pstat.tm` Property back to 0 (FALSE). If the Initiate Secure Software Update bit of  
4486 `/oic/sec/pstat.tm` is set to 0 (FALSE) by a Client, it has no effect on the update process.

#### 4487 **14.5.5 Recommended Usage**

4488 The Initiate Secure Software Update bit of `/oic/sec/pstat.tm` should only be set by a Client after the  
4489 Initiate Software Version Validation check is complete.

4490 The process of updating Device software may involve state changes that affect the Device  
4491 Operational State (`/oic/sec/pstat.dos`). Devices with an interest in the Device(s) being updated

4492 should monitor /oic/sec/pstat.dos and be prepared for pending software update(s) to affect Device  
4493 state(s) prior to completion of the update.

4494 The Device itself may indicate that it is autonomously initiating a software version check/update or  
4495 that a check/update is complete by setting the pstat.tm and pstat.cm Initiate Software Version  
4496 Validation and Secure Software Update bits when starting or completing the version check or  
4497 update process. As is the case with a Client-initiated update, Clients can be notified that an  
4498 autonomous version check or software update is pending and/or complete by observing pstat  
4499 resource changes.

#### 4500 **14.6 Non-OCF Endpoint interoperability**

#### 4501 **14.7 Security Levels**

4502 Security Levels are a way to differentiate Devices based on their security criteria. This need for  
4503 differentiation is based on the requirements from different verticals such as industrial and health  
4504 care and may extend into smart home. This differentiation is distinct from Device classification  
4505 (e.g. IETF RFC 7228)

4506 These categories of security differentiation may include, but is not limited to:

- 4507 1) Security Hardening
- 4508 2) Identity Attestation
- 4509 3) Certificate/Trust
- 4510 4) Onboarding Technique
- 4511 5) Regulatory Compliance
- 4512 a) Data at rest
- 4513 b) Data in transit
- 4514 6) Cipher Suites – Crypto Algorithms & Curves
- 4515 7) Key Length
- 4516 8) Secure Boot/Update

4517 In the future security levels can be used to define interoperability.

4518 The following applies to the OCF Security Specification 1.1:

4519 The current document does not define any other level beyond Security Level 0. All Devices will be  
4520 designated as Level 0. Future versions may define additional levels.

4521 Additional comments:

- 4522 – The definition of a given security level will remain unchanged between versions of the document.
- 4523 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 4524 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of the  
4525 higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and a later  
4526 document version defines a security level 1, the Device could be evaluated and classified as  
4527 level 1 if it meets level 1 requirements).
- 4528 – The security levels may need to be visible to the end user.

#### 4529 **14.8 Security Profiles**

##### 4530 **14.8.1 Security Profiles General**

4531 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This need  
4532 for differentiation is based on the requirements from different verticals such as industrial and health

4533 care and may extend into smart home. This differentiation is distinct from device classification (e.g.  
4534 IETF RFC 7228)

4535 These categories of security differentiation may include, but is not limited to:

- 4536 1) Security Hardening and assurances criteria
- 4537 2) Identity Attestation
- 4538 3) Certificate/Trust
- 4539 4) Onboarding Technique
- 4540 5) Regulatory Compliance
- 4541 a) Data at rest
- 4542 b) Data in transit
- 4543 6) Cipher Suites – Crypto Algorithms & Curves
- 4544 7) Key Length
- 4545 8) Secure Boot/Update

4546 Each Security Profile definition must specify the version or versions of the OCF Security  
4547 Specification(s) that form a baseline set of normative requirements. The profile definition may  
4548 include security requirements that supersede baseline requirements (not to relax security  
4549 requirements).

4550 Security Profiles have the following properties:

- 4551 – A given profile definition is not specific to the version of the document that defines it. For  
4552 example, the profile may remain constant for subsequent OCF Security Specification versions.
- 4553 – A specific OCF Device and platform combination may be used to satisfy the security profile.
- 4554 – Profiles may have overlapping criteria, hence it may be possible to satisfy multiple profiles  
4555 simultaneously.
- 4556 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found to  
4557 satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the document,  
4558 and a later document version defines a security profile Black, the device could be evaluated and  
4559 classified as profile Black if it meets profile Black requirements).
- 4560 – A machine-readable representation of compliance results specifically describing profiles  
4561 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or  
4562 manifest may contain security profiles attributes).

#### 4563 **14.8.2 Identification of Security Profiles (Normative)**

##### 4564 **14.8.2.1 Security Profiles in Prior Documents**

4565 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles  
4566 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in  
4567 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use  
4568 the OCF Security Specification version to characterize expected security behavior.

##### 4569 **14.8.2.2 Security Profile Resource Definition**

4570 The oic.sec.sp Resource is used by the OCF Device to show which OCF Security Profiles the OCF  
4571 Device is capable of supporting and which are authorized for use by the OCF Security Domain  
4572 owner. Properties of the Resource identify which OCF Security Profile is currently operational. The  
4573 ocfSecurityProfileOID value type shall represent OID values and may reference an entry in the form  
4574 of strings (UTF-8).

4575 "oic.sec.sp" Resource is defined in Table 76.

4576

**Table 73 – Definition of the oic.sec.sp Resource**

Fixed URI	Resource Type Title	Resource Type ID ("rt" value)	OCF Interfaces	Description	Related Functional Interaction
/oic/sec/sp	Security Profile Resource Definition	oic.r.sp	oic.if.baseline	Resource specifying supported and current security profile(s)	Discoverable

4577 Table 77 defines the Properties of "oic.sec.sp".

4578

**Table 74 – Properties of the oic.sec.sp Resource**

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandatory	Description
Supported Security Profiles	supportedprofiles	ocfSecurityProfileOID	array	RW	Yes	Array of supported Security Profiles (e.g. ["1.3.6.1.4.1.51414.0.0.2.0","1.3.6.1.4.1.51414.0.0.3.0"])
SecurityProfile	currentprofile	ocfSecurityProfileOID	N/A	RW	Yes	Currently active Security Profile (e.g. "1.3.6.1.4.1.51414.0.0.3.0")

4579 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or  
 4580 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

```

4581 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
4582     private(4) enterprise(1) OCF(51414) }
4583
4584 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
4585
4586 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
4587
4588     sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
4589     --The Security Profile is not specified
4590     sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
4591     --This specifies the OCF Baseline Security Profile(s)
4592     sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
4593     --This specifies the OCF Black Security Profile(s)
4594     sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
4595     --This specified the OCF Blue Security Profile(s)
4596     sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
4597     --This specifies the OCF Purple Security Profile(s)
4598
4599     --versioned Security Profiles
4600     sp-unspecified-v0 ::= ocfSecurityProfileOID {id-sp-unspecified 0}
4601     --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"
4602     sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
4603     --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"
4604     sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
4605     --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"
4606     sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
4607     --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"
4608     sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
4609     --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"
4610
4611     ocfSecurityProfileOID ::= UTF8String
4612
  
```

### 4613 14.8.3 Security Profiles

#### 4614 14.8.3.1 Security Profiles General

4615 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to  
 4616 the Security Profile clauses for additional details).

4617 The OCF Conformance criteria may require vendor attestation that establishes the expected  
4618 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific  
4619 requirements).

#### 4620 **14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)**

4621 The Security Profile "sp-unspecified-v0" is reserved for future use.

#### 4622 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4623 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where  
4624 the /oic/sec/sp Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the  
4625 "supportedprofiles" Property of the /oic/sec/sp Resource.

4626 It indicates the OCF Device satisfies the normative security requirements for this document.

4627 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-  
4628 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other  
4629 profiles.

4630 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to  
4631 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

#### 4632 **14.8.3.4 Security Profile Black (sp-black-v0)**

##### 4633 **14.8.3.4.1 Black Profile General**

4634 The need for Security Profile Black v0 is to support devices and manufacturers who wish to certify  
4635 their devices meeting this specific set of security criteria. A Device may satisfy the Black  
4636 requirements as well as requirements of other profiles, the Black Security Profile is not necessarily  
4637 mutually exclusive with other Security Profiles unless those requirements conflict with the explicit  
4638 requirements of the Black Security Profile.

##### 4639 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4640 Security Profile Black devices could include any device a manufacturer wishes to certify at this  
4641 profile, but healthcare devices and industrial devices with additional security requirements are the  
4642 initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or devices  
4643 with exceptional profiles of trust bestowed upon them, may wish to certify at this profile; these types  
4644 of devices may include, but are not limited to the following:

- 4645 – Bridges (Mapping devices between ecosystems handling virtual devices from different  
4646 ecosystems)
- 4647 – Resource Directories (Devices trusted to manage OCF Security Domain resources)
- 4648 – Remote Access (Devices which have external access but can also act within the OCF Security  
4649 Domain)
- 4650 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)
- 4651 – Industrial Devices (Devices with advanced management, security and attestation requirements)

##### 4652 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4653 Every device with "currentprofile" Property of the /oic/sec/sp Resource designating a Security  
4654 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4655 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4656 – Support for AES 128 encryption for data at rest and in transit.
- 4657 – Hardening minimums: manufacturer assertion of secure credential storage

4658 – In 7.1.2 in enumerated item #2: “The value should correspond to the value of subjectPublicKey  
4659 defined in SubjectPublicKeyInfo of the certificate” is changed to require this format: “The value  
4660 shall correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo of the  
4661 certificate.”

4662 – In 7.1.2 in the enumerated item #3: “The value should correspond to the value of  
4663 subjectPublicKey defined in SubjectPublicKeyInfo” is changed to require this format: “The value  
4664 SHALL correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo.”

4665 – In 14) in enumerated item #10 “The /oic/sec/cred Resource should contain credential(s) if  
4666 required by the selected OTM” is changed to require the credential be stored: “The /oic/sec/cred  
4667 Resource shall contain credential(s).”

4668 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4669 certificate and the extension’s ‘securityProfile’ field shall contain sp-black-v0 represented by  
4670 the ocfSecurityProfileOID string, “1.3.6.1.4.1.51414.0.0.2.0”.

4671 When a device supports the black profile, the “supportedprofiles” Property shall contain sp-black-  
4672 v0, represented by the OID string “1.3.6.1.4.1.51414.0.0.2.0”, and may contain other profiles.

4673 When a manufacturer makes sp-black-v0 the default, by setting the “currentprofile” Property to  
4674 “1.3.6.1.4.1.51414.0.0.2.0”, the “supportedprofiles” Property shall contain sp-black-v0.

4675 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework  
4676 described in the supporting documents:

4677 – Certificate Profile (See 9.4.2)

4678 – Certificate Policy (see Certificate Policy document: OCF-TSC-SWG-CP-D03-171101.docx)

4679 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4680 **14.8.3.5.1 Blue Profile General**

4681 The Security Profile Blue is used when manufacturers issue platform certificates for platforms  
4682 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is  
4683 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF  
4684 Security Domain owners evaluate manufacturer supplied certificates and attributed data to  
4685 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.  
4686 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may  
4687 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4688 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting  
4689 Criteria defined by OCF.

4690 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4691 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from  
4692 OCF Device vendor and where platform vendors may implement trusted platforms that may conform  
4693 to industry standards defining trusted platforms. The OCF Security Profile Blue specifies  
4694 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance criteria  
4695 produced by OCF conformance operations. The OCF Security Domain owner evaluates these data  
4696 when an OCF Device is onboarded into the OCF Security Domain. Based on this evaluation the  
4697 OCF Security Domain owner determines which Security Profile may be applied during OCF Device  
4698 operation. All OCF Device types may be considered for evaluation using the OCF Security Profile  
4699 Blue.

4700 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4701 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for this  
4702 document version are satisfied and the following additional criteria are satisfied.

- 4703 OCF Blue profile defines the following OCF Device quality assurances:
- 4704 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF Device  
4705 was hosted on one or more platforms that satisfies OCF Blue platform security assurances and  
4706 platform security and privacy functionality requirements.
  - 4707 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF and  
4708 published by OCF in a machine readable format.
  - 4709 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned  
4710 signing key.
  - 4711 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4712 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by the  
4713 ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
  - 4714 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in  
4715 its certificate.
  - 4716 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF  
4717 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's  
4718 'securityprofiles' field.
- 4719 OCF Blue profile defines the following OCF Device security functionality:
- 4720 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage  
4721 functions are hardened by the platform.
  - 4722 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials using  
4723 the /oic/sec/cred Resource where the 'credusage' Property contains the value  
4724 "oic.sec.cred.mfgcert".
  - 4725 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an IEEE802.1AR  
4726 IDevID and should verify the "TCG Endorsement Key Credential". All TCG-defined manufacturer  
4727 credentials may be identified by the "oic.sec.cred.mfgcert" value of the 'credusage' Property of  
4728 the /oic/sec/cred Resource. They may be used in response to selection of the  
4729 "oic.sec.doxm.mfgcert" owner transfer method.
  - 4730 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See  
4731 NIST SP 800-57).
  - 4732 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST SP  
4733 800-57).
  - 4734 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST  
4735 SP 800-57).
  - 4736 – OCF Device(s) should protect the /oic/sec/cred resource using the platform provided secure  
4737 storage.
  - 4738 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned  
4739 certificates) using platform provided secure storage.
  - 4740 – OCF Device(s) should check certificate revocation status for locally issued certificates.
  - 4741 – OCF onboarding tools (aka DOTS) shall check certificate revocation status for all certificates in  
4742 manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation fails  
4743 and the connection is refused. The DOTS may disregard revocation status results if unavailable.
- 4744 OCF Blue profile defines the following platform security assurances:
- 4745 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage  
4746 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL  
4747 Level 2.

- 4748 – Platforms implementing trusted platform functionality should be evaluated with a minimum  
4749 Common Criteria EAL Level 1.
- 4750 OCF Blue profile defines the following platform security and privacy functionality:
- 4751 – The Platform shall implement cryptographic service provider (CSP) functionality.
- 4752 – Platform CSP functionality shall include cryptographic algorithms, random number generation,  
4753 secure time.
- 4754 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST SP  
4755 800-57).
- 4756 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See  
4757 NIST SP 800-57).
- 4758 – Platforms hosting OCF Device(s) should implement a platform identifier following IEEE802.1AR  
4759 or Trusted Computing Group(TCG) specifications.
- 4760 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF Device(s)  
4761 should supply TCG-defined manufacture certificates; also known as "TCG Endorsement Key  
4762 Credential" (which complies with IETF RFC 5280) and "TCG Platform Credential" (which  
4763 complies with IETF RFC 5755).
- 4764 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,  
4765 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.
- 4766 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to  
4767 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.
- 4768 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"  
4769 Property to one of the other values found in the "supportedprofiles" Property.
- 4770 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**
- 4771 Every device with the /oic/sec/sp Resource designating "sp-purple-v0", as defined in clause 14.8.2  
4772 must support following minimum requirements
- 4773 – Hardening minimums: secure credential storage, software integrity validation, secure update.
- 4774 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension  
4775 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-  
4776 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"
- 4777 – The OCF Device shall include a X.509v3 OCF CPLAttributes Extension (clause 9.4.2.2.7) in its  
4778 End-Entity Certificate when manufacturer certificate is used.
- 4779 Security Profile Purple has following optional security hardening requirements that the device can  
4780 additionally support.
- 4781 – Hardening additions: secure boot, hardware backed secure storage
- 4782 – The OCF Device shall include a X.509v3 OCF SecurityClaims Extension (clause 9.4.2.2.6) in its  
4783 End-Entity Certificate and it shall include corresponding OIDs to the hardening additions  
4784 implemented and attested by the vendor. If there is no additional support for hardening  
4785 requirements, X.509v3 OCF SecurityClaims Extension shall be omitted.
- 4786 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism  
4787 for security critical executables such as cryptographic modules or secure service applications, and  
4788 they should be validated before the execution. The key used for validating the integrity must be  
4789 pinned at the least to the validating software module.
- 4790 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4791 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM) to  
4792 be executed by the processor on power-on, and secure boot parameters to be provisioned by  
4793 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the  
4794 security critical executables and stop the boot process if any integrity of them is compromised.

4795 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile  
4796 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic  
4797 attacks.

4798 More details on security hardening guidelines for software integrity validation, secure boot, secure  
4799 update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4800 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA Vetting  
4801 Criteria defined by OCF.

4802 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-purple-  
4803 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other profiles.

4804 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to  
4805 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

4806 **15 Device Type Specific Requirements**

4807 **15.1 Bridging Security**

4808 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4809 The OCF Bridge Device shall go through OCF ownership transfer as any other onboarder would.

4810 The software of an OCF Bridge Device shall be field updatable. (This requirement need not be  
4811 tested but can be certified via a vendor declaration.)

4812 Each Virtual OCF Device shall be onboarded by an OCF Onboarding tool. Each Virtual Bridged  
4813 Device should be provisioned as appropriate in the Bridged Protocol. In other words, Virtual OCF  
4814 Devices and Virtual Bridged Devices are treated the same way as physical Devices. They are  
4815 entities that have to be provisioned in their network.

4816 Each Virtual OCF Device shall implement the behaviour required by ISO/IEC 30118-1:2018 and  
4817 this document. Each Virtual OCF Device shall perform authentication, access control, and  
4818 encryption according to the security settings it received from the Onboarding Tool. Each Virtual  
4819 Bridged Device shall implement the security requirements of the Bridged Protocol.

4820 In addition, in order to be considered secure from an OCF perspective, the OCF Bridge Device  
4821 implementation shall use appropriate ecosystem-specific security options for communication  
4822 between the Virtual Bridged Devices instantiated by the OCF Bridge Device and Bridged Devices.  
4823 This security shall include mutual authentication, and encryption and integrity protection of  
4824 messages in the bridged ecosystem.

4825 A Virtual OCF Device may authenticate itself to the DOTS using the Manufacturer Certificate Based  
4826 OTM (see clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the OCF  
4827 Bridge Device which instantiated that Virtual OCF Device.

4828 A Virtual OCF Device may authenticate itself to the OCF Cloud (see clause 10.5.2) using the  
4829 Manufacturer Certificate and corresponding private key of the OCF Bridge Device which  
4830 instantiated that Virtual OCF Device.

4831 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4832 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4833 For AllJoyn translator, an OCF Onboarding Tool shall be able to block the communication of all  
4834 OCF Devices with all Bridged Devices that don't communicate securely with the Bridge, by using  
4835 the Bridge Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

4836 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4837 The OCF Bridge Device shall implement oneM2M application access control as defined in the  
4838 oneM2M Release 3 Specifications

4839 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices  
4840 that don't communicate securely with the OCF Bridge Device

4841 **15.1.2.3 Additional Security Requirements specific to the U+ Protocol**

4842 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices  
4843 that don't communicate securely with the OCF Bridge Device.

4844 **15.1.2.4 Additional Security Requirements specific to the Z-Wave Protocol**

4845 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices  
4846 that don't communicate securely with the OCF Bridge Device.

4847 **15.1.2.5 Additional Security Requirements specific to the Zigbee Protocol**  
4848 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices  
4849 that don't communicate securely with the OCF Bridge Device.  
4850  
4851  
4852  
4853  
4854  
4855  
4856  
4857  
4858  
4859  
4860  
4861  
4862  
4863  
4864  
4865  
4866  
4867  
4868  
4869  
4870  
4871 .

4872  
4873  
4874

## Annex A (informative) Access Control Examples

4875

### A.1 Example OCF ACL Resource

4876 Figure A-1 shows how a /oic/sec/acl2 Resource could be configured to enforce an example access  
4877 policy on the Server.

```
4878 {
4879   "aclist2": [
4880     {
4881       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve, Update,
4882       Delete and Notify)
4883       "subject": {"uuid": "XXX-...-XX01"},
4884       "resources": [
4885         {"href": "/oic/sh/light/1"},
4886         {"href": "/oic/sh/temp/0"}
4887       ],
4888       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC
4889       "validity": [
4890         // The period starting at 18:00:00 UTC, on January 1, 2015 and
4891         // ending at 07:00:00 UTC on January 2, 2015
4892         "period": ["20150101T180000Z/20150102T070000Z"],
4893         // Repeats the {period} every week until the last day of Jan. 2015.
4894         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
4895       ],
4896       "aceid": 1
4897     }
4898   ],
4899   // An ACL provisioning and management service should be identified as
4900   // the resource owner
4901   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
4902 }
4903
```

4903 **Figure A-1 – Example /oic/sec/acl2 Resource**

4904

### A.2 Example AMS

4905 Figure A-2 demonstrates how the /oic/sec/amacl Resource should be configured to achieve this  
4906 objective.

```
4907 {
4908   "resources": [
4909     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
4910     // supplied then use the sacl validation credential to enforce access.
4911     {"href": "/oic/sh/light/1"},
4912     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
4913     // supplied then use the sacl validation credential to enforce access.
4914     {"href": "/oma/3"},

```

```
4915 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4916 // the sacl validation credential to enforce access.
4917 {"wc": "**"}
4918 }
4919
```

**Figure A-2 Example /oic/sec/amacl Resource**

4920  
4921  
4922

**Annex B  
(Informative)  
Execution Environment Security Profiles**

4923 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security  
4924 robustness requirements meeting all IOT applications and services will not serve the needs of OCF,  
4925 and security profiles of varying degree of robustness (trustworthiness), cost and complexity have  
4926 to be defined. To address a large ecosystem of vendors, the profiles can only be defined as  
4927 requirements and the exact solutions meeting those requirements are specific to the vendors' open  
4928 or proprietary implementations, and thus in most part outside scope of this document.

4929 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228  
4930 (Terminology for constrained node networks) methodology, we limit the number of security profiles  
4931 to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities criteria for  
4932 each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4933 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are  
4934 either capable of no security functionality or easily breakable security that depend on environmental  
4935 (e.g. availability of human) factors to perform security functions. This means the class 0 will not be  
4936 equipped with an SEE.

4937 **Table B.1 – OCF Security Profile**

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4938 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of  
4939 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,  
4940 memory, storage).

Annex C  
(normative)  
Resource Type definitions

**C.1 List of Resource Type definitions**

Table C.1 contains the list of defined security resources in this document.

**Table C.1 – Alphabetized list of security resources**

Friendly Name (informative)	Resource Type (rt)	Clause
Access Control List	oic.r.acl	C.3
Access Control List 2	oic.r.acl2	C.4
Account	oic.r.account	C.2
Account Session	oic.r.session	C.13
Account Token Refresh	oic.r.tokenrefresh	C.15
Certificate Revocation	oic.r.crl	C.7
Certificate Signing Request	oic.r.crl	C.8
Credential	oic.r.cred	C.6
Device owner transfer method	oic.r.doxm	C.9
Device Provisioning Status	oic.r.pstat	C.10
Managed Access Control	oic.r.acl2	C.5
Roles	oic.r.pstat	C.11
Security Profile	oic.r.sp	C.14
Signed Access Control List	oic.r.sacl	C.12

Commented [OA5]: Remove this please

**C.2 Account Token**

**C.2.1 Introduction**

Sign-up using generic account provider.

**C.2.2 Well-known URI**

/oic/sec/account

**C.2.3 Resource type**

The resource type (rt) is defined as: ['oic.r.account'].

**C.2.4 OpenAPI 2.0 definition**

```
{
  "swagger": "2.0",
  "info": {
    "title": "Account Token",
    "version": "20190111",
    "license": {
      "name": "OCF Data Model License",
      "url":
        "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
        CENSE.md",
      "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
        reserved."
    }
  }
}
```

```

4970     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
4971   },
4972   "schemes": ["http"],
4973   "consumes": ["application/json"],
4974   "produces": ["application/json"],
4975   "paths": {
4976     "/oic/sec/account" : {
4977       "post": {
4978         "description": "Sign-up using generic account provider.\n",
4979         "parameters": [
4980           {"$ref": "#/parameters/interface"},
4981           {
4982             "name": "body",
4983             "in": "body",
4984             "required": true,
4985             "schema": { "$ref": "#/definitions/Account-request" },
4986             "x-example": {
4987               "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
4988               "authprovider" : "github",
4989               "accesstoken" : "8802f2eaf8b5e147a936"
4990             }
4991           }
4992         ],
4993         "responses": {
4994           "204": {
4995             "description": "2.04 Changed respond with required and optional information\n",
4996             "x-example": {
4997               "rt": ["oic.r.account"],
4998               "accesstoken" : "0f3d9f7fe5491d54077d",
4999               "refreshToken" : "00fe4644a6f8b5e147a936",
5000               "expiresin" : 3600,
5001               "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
5002               "redirecturi" : "coaps+tcp://example.com:443"
5003             },
5004             "schema": { "$ref": "#/definitions/Account-response" }
5005           }
5006         }
5007       }
5008     },
5009     "delete": {
5010       "description": "Delete a device. This also removes all resources in the device on cloud
5011 side.\nexample: /oic/account?di=9cfbeb8e-5ale-4dlc-9d01-
5012 00c04fd430c8&accesstoken=0f3d9f7fe5491d54077d\n",
5013       "parameters": [
5014         {"$ref": "#/parameters/interface"}
5015       ],
5016       "responses": {
5017         "202": {
5018           "description": "2.02 Deleted response informing the device is successfully
5019 deleted.\n"
5020         }
5021       }
5022     }
5023   }
5024 },
5025 "parameters": {
5026   "interface": {
5027     "in": "query",
5028     "name": "if",
5029     "type": "string",
5030     "enum": ["oic.if.baseline"]
5031   }
5032 },
5033 "definitions": {
5034   "Account-request": {
5035     "properties": {
5036       "authprovider": {
5037         "description": "The name of Authorization Provider through which Access Token was
5038 obtained",
5039         "type": "string"
5040       }

```

```

5041     },
5042     "accesstoken" : {
5043       "description": "Access-Token used for communication with OCF Cloud after account
5044 creation",
5045       "pattern": "(?!$|\\s+).*",
5046       "type": "string"
5047     },
5048     "di": {
5049       "description": "Format pattern according to IETF RFC 4122.",
5050       "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5051 9]{12}$",
5052       "type": "string"
5053     },
5054   },
5055   "type" : "object",
5056   "required": ["di", "accesstoken"]
5057 },
5058 "Account-response": {
5059   "properties": {
5060     "expiresin" : {
5061       "description": "Access-Token remaining life time in seconds (-1 if permanent)",
5062       "readOnly": true,
5063       "type": "integer"
5064     },
5065     "rt": {
5066       "description": "Resource Type of the Resource",
5067       "items": {
5068         "maxLength": 64,
5069         "type": "string",
5070         "enum" : ["oic.r.account"]
5071       },
5072       "minItems": 1,
5073       "maxItems": 1,
5074       "readOnly": true,
5075       "type": "array"
5076     },
5077     "refreshtoken" : {
5078       "description": "Refresh token can be used to refresh the Access Token before getting
5079 expired",
5080       "pattern": "(?!$|\\s+).*",
5081       "readOnly": true,
5082       "type": "string"
5083     },
5084     "uid" : {
5085       "description": "Format pattern according to IETF RFC 4122.",
5086       "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5087 9]{12}$",
5088       "type": "string"
5089     },
5090     "accesstoken" : {
5091       "description": "Access-Token used for communication with cloud after account creation",
5092       "pattern": "(?!$|\\s+).*",
5093       "type": "string"
5094     },
5095     "n": {
5096       "$ref":
5097 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5098 schema.json#/definitions/n"
5099     },
5100     "id": {
5101       "$ref":
5102 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5103 schema.json#/definitions/id"
5104     },
5105     "redirecturi" : {
5106       "description": "Using this URI, the Client needs to reconnect to a redirected OCF Cloud.
5107 If provided, this value shall be used by the Device instead of Mediator-provided URI during the
5108 Device Registration.",
5109       "readOnly": true,
5110       "type": "string"
5111     },

```

```

5112     "if": {
5113         "description": "The interface set supported by this resource",
5114         "items": {
5115             "enum": [
5116                 "oic.if.baseline"
5117             ],
5118             "type": "string"
5119         },
5120         "minItems": 1,
5121         "maxItems": 1,
5122         "uniqueItems": true,
5123         "readOnly": true,
5124         "type": "array"
5125     },
5126 },
5127 "type" : "object",
5128 "required": ["accesstoken", "refresh token", "expiresin", "uid"]
5129 }
5130 }
5131 }
5132

```

### 5133 C.2.5 Property definition

5134 Table C.2 defines the Properties that are part of the ['oic.r.account'] Resource Type

5135 **Table C.2 – The Property definitions of the Resource with type 'rt' = ['oic.r.account']**

Property name	Value type	Mandatory	Access mode	Description
refresh token	string	Yes	Read Only	Refresh token can be used to refresh the Access Token before getting expired
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
accesstoken	string	Yes	Read Write	Access-Token used for communication with cloud after account creation
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds (-1 if permanent)
rt	array: see schema	No	Read Only	Resource Type of the Resource
redirecturi	string	No	Read Only	Using this URI, the Client needs to reconnect to a redirected OCF Cloud. If provided, this value shall be used by the Device instead of Mediator-provided URI during the

				Device Registration.
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
authprovider	string	No	Read Write	The name of Authorization Provider through which Access Token was obtained
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
accesstoken	string	Yes	Read Write	Access-Token used for communication with OCF Cloud after account creation

5136 **C.2.6 CRUDN behaviour**

5137 Table C.3 defines the CRUDN operations that are supported on the ['oic.r.account'] Resource Type

5138 **Table C.3 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.account']**

Create	Read	Update	Delete	Notify
		post	delete	

5139 **C.3 Access Control List - DEPRECATED**

5140 **C.3.1 Introduction**

5141 This Resource specifies the local access control list.  
 5142 When used without query parameters, all the ACE entries are returned.  
 5143 When used with a subjectuid, only the ACEs with the specified  
 5144 subjectuid are returned. If subjectuid and resources are specified,  
 5145 only the ACEs with the specified subjectuid and Resource hrefs are  
 5146 returned.

5148 **C.3.2 Well-known URI**

5149 /oic/sec/acl

5150 **C.3.3 Resource type**

5151 The resource type (rt) is defined as: ['oic.r.acl'].

5152 **C.3.4 OpenAPI 2.0 definition**

5153 {  
 5154 "swagger": "2.0",  
 5155 "info": {  
 5156 "title": "Access Control List",  
 5157 "version": "v1.1-20161213",  
 5158 "license": {  
 5159 "name": "OCF Data Model License",

Commented [OA6]: Remove if we don't care about clause numbering

```

5160 ----- "url":
5161 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e03661e4e0fbce8bdc4ba/Lf
5162 CENCE.md",
5163 ----- "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5164 reserved."
5165 ----- }
5166 ----- "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5167 ----- }
5168 ----- "schemes": ["http"],
5169 ----- "consumes": ["application/json"],
5170 ----- "produces": ["application/json"],
5171 ----- "paths": {
5172 ----- "/oic/sec/acl": {
5173 ----- "get": {
5174 ----- "description": "This Resource specifies the local access control list.\nWhen used without
5175 query parameters, all the ACE entries are returned.\nWhen used with a subjectuid, only the ACEs
5176 with the specified\nsubjectuid are returned. If subjectuid and resources are specified,\nonly the
5177 ACEs with the specified subjectuid and Resource hrefs are\nreturned.\n",
5178 ----- "parameters": {
5179 ----- {"$ref": "#/parameters/interface"},
5180 ----- {"$ref": "#/parameters/ace-filtered-uid"},
5181 ----- {"$ref": "#/parameters/ace-filtered-resources"}
5182 ----- },
5183 ----- "responses": {
5184 ----- "200": {
5185 ----- "description": "",
5186 ----- "x-example":
5187 ----- {
5188 ----- "rt": ["oic.r.acl"],
5189 ----- "aclist": {
5190 ----- "aces": {
5191 ----- {
5192 ----- "subjectuid": "e61c3e6b-9c54-4b81-8ce5-f9039cid04d9",
5193 ----- "resources": {
5194 ----- {
5195 ----- "href": "coaps://IP_ADDR/temp",
5196 ----- "rel": "some-rel",
5197 ----- "rt": ["oic.r.temperature"],
5198 ----- "if": ["oic.if.a"]
5199 ----- }
5200 ----- }
5201 ----- {
5202 ----- "href": "coaps://IP_ADDR/temp",
5203 ----- "rel": "some-rel",
5204 ----- "rt": ["oic.r.temperature"],
5205 ----- "if": ["oic.if.s"]
5206 ----- }
5207 ----- }
5208 ----- "permission": 31,
5209 ----- "validity": {
5210 ----- {
5211 ----- "period": "20160101T180000Z/20170102T070000Z",
5212 ----- "recurrence": { "DSTART:XXXX",
5213 ----- "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" }
5214 ----- }
5215 ----- }
5216 ----- "period": "20160101T180000Z/PT5H30M",
5217 ----- "recurrence": { "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" }
5218 ----- }
5219 ----- }
5220 ----- }
5221 ----- }
5222 ----- "rowneruid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5223 ----- }
5224 ----- "schema": { "$ref": "#/definitions/Acl" }
5225 ----- }
5226 ----- "400": {
5227 ----- "description": "The request is invalid."
5228 ----- }
5229 ----- }
5230 ----- }

```

```

5231 ----- "post": {
5232 -----   "description": "Updates the ACL resource with the provided values. ACEs provided\nin the
5233 update not currently in the ACL are added. ACEs that already\nexist in the ACL are ignored.\n\nNote
5234 that for the purposes of update, equivalency is determined\nby comparing the ACE subjectuid,
5235 permission, string comparisons\nof all validity elements, and string comparisons of all
5236 resource\nhrefs.\n",
5237 -----   "parameters": {
5238 -----     {"$ref": "#/parameters/interface"},
5239 -----     {
5240 -----       "name": "body",
5241 -----       "in": "body",
5242 -----       "required": true,
5243 -----       "schema": { "$ref": "#/definitions/acl" },
5244 -----       "x-example":
5245 -----       {
5246 -----         "aclist": {
5247 -----           "aces": [
5248 -----             {
5249 -----               "subjectuid": "e61e3e6b-9e54-4b81-8ee5-f9039eld04d9",
5250 -----               "resources": [
5251 -----                 {
5252 -----                   "href": "coaps://IP-ADDR/temp",
5253 -----                   "rel": "some-rel",
5254 -----                   "rt": ["oic.r.temperature"],
5255 -----                   "if": ["oic.if.a"]
5256 -----                 },
5257 -----                 {
5258 -----                   "href": "coaps://IP-ADDR/temp",
5259 -----                   "rel": "some-rel",
5260 -----                   "rt": ["oic.r.temperature"],
5261 -----                   "if": ["oic.if.s"]
5262 -----                 }
5263 -----               ],
5264 -----               "permission": 31,
5265 -----               "validity": {
5266 -----                 {
5267 -----                   "period": "20160101T180000Z/20170102T070000Z",
5268 -----                   "recurrence": { "DSTART:XXXX",
5269 -----                     "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" }
5270 -----                 },
5271 -----                 {
5272 -----                   "period": "20160101T180000Z/PT5H30M",
5273 -----                   "recurrence": { "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" }
5274 -----                 }
5275 -----               ]
5276 -----             }
5277 -----           ],
5278 -----           "rowneruid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5279 -----         }
5280 -----       }
5281 -----     },
5282 -----   },
5283 -----   "responses": {
5284 -----     "400": {
5285 -----       "description": "The request is invalid."
5286 -----     },
5287 -----     "201": {
5288 -----       "description": "The ACL entry/entries is/are created."
5289 -----     },
5290 -----     "204": {
5291 -----       "description": "The ACL entry/entries is/are updated."
5292 -----     }
5293 -----   },
5294 ----- },
5295 ----- "delete": {
5296 -----   "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5297 ACE entries are deleted.\nWhen DELETE is used with a subjectuid, only the ACEs with the
5298 specified\nsubjectuid are deleted. If subjectuid and resource hrefs are specified,\nonly the ACEs with
5299 the specified subjectuid and resource hrefs are\ndeleted.\n",
5300 -----   "parameters": {
5301 -----     {"$ref": "#/parameters/interface"},

```

```

5302     { "$ref": "#/parameters/ace-filtered-uuid"},
5303     { "$ref": "#/parameters/ace-filtered-resources"}
5304   },
5305   "responses": {
5306     "200": {
5307       "description": "The matching ACEs or the entire ACL resource has been successfully
5308 deleted."
5309     },
5310     "400": {
5311       "description": "The request is invalid."
5312     }
5313   }
5314 }
5315 }
5316 },
5317 "parameters": {
5318   "interface": {
5319     "in": "query",
5320     "name": "if",
5321     "type": "string",
5322     "enum": {"oic.if.baseline"}
5323   },
5324   "ace-filtered-uuid": {
5325     "in": "query",
5326     "name": "subjectuuid",
5327     "required": false,
5328     "type": "string",
5329     "description": "Only applies to ACEs with the specified subject UUID",
5330     "x-example": "ee61e3e6b-9e54-4b81-8ee5-f9039e1d04d9"
5331   },
5332   "ace-filtered-resources": {
5333     "in": "query",
5334     "name": "resources",
5335     "required": false,
5336     "type": "string",
5337     "description": "Only applies to ACEs with the specified subjectuuid | and resources href",
5338     "x-example": "oic://IP-ADDR/temp"
5339   }
5340 },
5341 "definitions": {
5342   "acl": {
5343     "properties": {
5344       "owneruuid": {
5345         "description": "The value identifies the unique resource owner\nFormat pattern according
5346 to IETF RFC 4122.",
5347         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5348 9]{12}$",
5349         "type": "string"
5350       },
5351       "rt": {
5352         "description": "Resource Type of the Resource",
5353         "items": {
5354           "maxLength": 64,
5355           "type": "string",
5356           "enum": {"oic.r.acl"}
5357         }
5358       },
5359       "readOnly": true,
5360       "type": "array"
5361     }
5362   },
5363   "aclist": {
5364     "description": "Subject-based Access Control Entries in the ACL resource",
5365     "properties": {
5366       "aces": {
5367         "items": {
5368           "permission": {
5369             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask
5370 indicating permissions",
5371             "x-detail-desc": {
5372               "0": "No permissions",

```

```

5373 ----- "1 - Create permission is granted",
5374 ----- "2 - Read, observe, discover permission is granted",
5375 ----- "4 - Write, update permission is granted",
5376 ----- "8 - Delete permission is granted",
5377 ----- "16 - Notify permission is granted"
5378 ----- },
5379 ----- "maximum": 31,
5380 ----- "minimum": 0,
5381 ----- "type": "integer"
5382 ----- },
5383 ----- "resources": {
5384 ----- "description": "References the application's resources to which a security
5385 policy applies",
5386 ----- "items": {
5387 ----- "properties": {
5388 ----- "anchor": {
5389 ----- "description": "This is used to override the context URI e.g. override the
5390 URI of the containing collection.",
5391 ----- "format": "uri",
5392 ----- "maxLength": 256,
5393 ----- "type": "string"
5394 ----- },
5395 ----- "di": {
5396 ----- "description": "The device ID\nFormat pattern according to IETF RFC
5397 4122.",
5398 ----- "pattern": "^[a-zA-F0-9]{8}-[a-zA-F0-9]{4}-[a-zA-F0-9]{4}-[a-zA-F0-9]{4}-
5399 {a-zA-F0-9}{12}$",
5400 ----- "type": "string"
5401 ----- },
5402 ----- "eps": {
5403 ----- "description": "the Endpoint information of the target Resource",
5404 ----- "items": {
5405 ----- "properties": {
5406 ----- "ep": {
5407 ----- "description": "Transport Protocol Suite + Endpoint Locator",
5408 ----- "format": "uri",
5409 ----- "type": "string"
5410 ----- },
5411 ----- "pri": {
5412 ----- "description": "The priority among multiple Endpoints",
5413 ----- "minimum": 1,
5414 ----- "type": "integer"
5415 ----- }
5416 ----- },
5417 ----- "type": "object"
5418 ----- },
5419 ----- "type": "array"
5420 ----- },
5421 ----- "href": {
5422 ----- "description": "This is the target URI, it can be specified as a Relative
5423 Reference or fully qualified URI.",
5424 ----- "format": "uri",
5425 ----- "maxLength": 256,
5426 ----- "type": "string"
5427 ----- },
5428 ----- "if": {
5429 ----- "description": "The interface set supported by this resource",
5430 ----- "items": {
5431 ----- "enum": [
5432 ----- "oic.if.baseline",
5433 ----- "oic.if.ll",
5434 ----- "oic.if.b",
5435 ----- "oic.if.rw",
5436 ----- "oic.if.x",
5437 ----- "oic.if.a",
5438 ----- "oic.if.s"
5439 ----- ],
5440 ----- "type": "string"
5441 ----- },
5442 ----- "minItems": 1,
5443 ----- "type": "array"

```

```

5444         }r
5445         "ins": {
5446             "description": "The instance identifier for this web link in an array of
5447 web links - used in collections",
5448             "type": "integer"
5449         }r
5450         "p": {
5451             "description": "Specifies the framework policies on the Resource
5452 referenced by the target URI",
5453             "properties": {
5454                 "bm": {
5455                     "description": "Specifies the framework policies on the Resource
5456 referenced by the target URI for e.g. observable and discoverable",
5457                     "type": "integer"
5458                 }
5459             },
5460             "required": [
5461                 "bm"
5462             ],
5463             "type": "object"
5464         }r
5465         "rel": {
5466             "description": "The relation of the target URI referenced by the link to
5467 the context URI",
5468             "oneOf": [
5469                 {
5470                     "default": {
5471                         "hosts"
5472                     },
5473                     "items": {
5474                         "maxLength": 64,
5475                         "type": "string"
5476                     },
5477                     "minItems": 1,
5478                     "type": "array"
5479                 },
5480                 {
5481                     "default": "hosts",
5482                     "maxLength": 64,
5483                     "type": "string"
5484                 }
5485             ]
5486         }r
5487         "xt": {
5488             "description": "Resource Type of the Resource",
5489             "items": {
5490                 "maxLength": 64,
5491                 "type": "string"
5492             },
5493             "minItems": 1,
5494             "type": "array"
5495         }r
5496         "title": {
5497             "description": "A title for the link relation. Can be used by the UI to
5498 provide a context.",
5499             "maxLength": 64,
5500             "type": "string"
5501         }r
5502         "type": {
5503             "default": "application/cbor",
5504             "description": "A hint at the representation of the resource referenced by
5505 the target URI. This represents the media types that are used for both accepting and emitting.",
5506             "items": {
5507                 "maxLength": 64,
5508                 "type": "string"
5509             },
5510             "minItems": 1,
5511             "type": "array"
5512         }
5513     }r
5514     "required": [

```

```

5515 _____ "href",
5516 _____ "xt",
5517 _____ "if"
5518 _____ },
5519 _____ "type": "object"
5520 _____ },
5521 _____ "type": "array"
5522 _____ },
5523 _____ "subjectuuid": {
5524 _____ "anyOf": [
5525 _____ {
5526 _____ "description": "The id of the device to which the ace applies to or \"*\
5527 for-anonymous-access",
5528 _____ "pattern": "^[\\*$]",
5529 _____ "type": "string"
5530 _____ },
5531 _____ ],
5532 _____ "description": "Format pattern according to IETF RFC 4122.",
5533 _____ "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5534 fA-F0-9]{12}$",
5535 _____ "type": "string"
5536 _____ }
5537 _____ },
5538 _____ },
5539 _____ "validity": {
5540 _____ "description": "validity is an array of time-pattern objects",
5541 _____ "items": {
5542 _____ "description": "The time-pattern contains a period and recurrence expressed in
5543 IETF RFC 5545 syntax",
5544 _____ "properties": {
5545 _____ "period": {
5546 _____ "description": "String represents a period using the IETF RFC 5545
5547 Period",
5548 _____ "type": "string"
5549 _____ },
5550 _____ "recurrence": {
5551 _____ "description": "String array represents a recurrence rule using the IETF
5552 RFC 5545 Recurrence",
5553 _____ "items": {
5554 _____ "type": "string"
5555 _____ },
5556 _____ "type": "array"
5557 _____ }
5558 _____ },
5559 _____ "required": [
5560 _____ "period"
5561 _____ ],
5562 _____ "type": "object"
5563 _____ },
5564 _____ "type": "array"
5565 _____ }
5566 _____ },
5567 _____ "required": [
5568 _____ "resources",
5569 _____ "permission",
5570 _____ "subjectuuid"
5571 _____ ],
5572 _____ "type": "object"
5573 _____ },
5574 _____ "type": "array"
5575 _____ }
5576 _____ },
5577 _____ "required": [
5578 _____ "aces"
5579 _____ ],
5580 _____ "type": "object"
5581 _____ },
5582 _____ "n": {
5583 _____ "$ref":
5584 _____ "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5585 schema.json#/definitions/n"

```

```

5586 }},
5587 "id": {
5588   "$ref":
5589   "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5590   schema.json#/definitions/id"
5591 },
5592 "if": {
5593   "description": "The interface set supported by this resource",
5594   "items": {
5595     "enum": [
5596       "oic.if.baseline"
5597     ]
5598   },
5599   "type": "string"
5600 },
5601 "minItems": 1,
5602 "readOnly": true,
5603 "type": "array"
5604 }
5605 "type": "object",
5606 "required": ["aclist", "rowneruuid"]
5607 }
5608 }
5609 }
5610

```

### 5611 C.3.5 — Property definition

5612 Table C.4 defines the Properties that are part of the ['oic.r.acl'] Resource Type

5613 **Table C.4 — The Property definitions of the Resource with type 'rt' = ['oic.r.acl']**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read-Write	The value identifies the unique resource owner. Format pattern according to IETF RFC 4122.
aclist	object: see schema	Yes	Read-Write	Subject-based Access Control Entries in the ACL resource
rt	array: see schema	No	Read-Only	Resource Type of the Resource
if	array: see schema	No	Read-Only	The OCF Interface set supported by this resource
id	multiple types: see schema	No	Read-Write	
n	multiple types: see schema	No	Read-Write	

### 5614 C.3.6 — CRUDN behaviour

5615 Table C.5 defines the CRUDN operations that are supported on the ['oic.r.acl'] Resource Type

5616 **Table C.5 — The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 5617 C.4 Access Control List-2

### 5618 C.4.1 Introduction

5619 This Resource specifies the local access control list.  
5620 When used without query parameters, all the ACE entries are returned.  
5621 When used with a query parameter, only the ACEs matching the specified  
5622 parameter are returned.  
5623

### 5624 C.4.2 Well-known URI

5625 /oic/sec/acl2

### 5626 C.4.3 Resource type

5627 The resource type (rt) is defined as: ['oic.r.acl2'].

### 5628 C.4.4 OpenAPI 2.0 definition

```
5629 {  
5630   "swagger": "2.0",  
5631   "info": {  
5632     "title": "Access Control List-2",  
5633     "version": "20190111",  
5634     "license": {  
5635       "name": "OCF Data Model License",  
5636       "url":  
5637         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
5638         CENSE.md",  
5639       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
5640         reserved."  
5641     },  
5642     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
5643   },  
5644   "schemes": ["http"],  
5645   "consumes": ["application/json"],  
5646   "produces": ["application/json"],  
5647   "paths": {  
5648     "/oic/sec/acl2" : {  
5649       "get": {  
5650         "description": "This Resource specifies the local access control list.\nWhen used without  
5651         query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs  
5652         matching the specified\nparameter are returned.\n",  
5653         "parameters": [  
5654           {"$ref": "#/parameters/interface"},  
5655           {"$ref": "#/parameters/ace-filtered"}  
5656         ],  
5657         "responses": {  
5658           "200": {  
5659             "description": "",  
5660             "x-example":  
5661               {  
5662                 "rt" : ["oic.r.acl2"],  
5663                 "aclist2": [  
5664                   {  
5665                     "aceid": 1,  
5666                     "subject": {  
5667                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",  
5668                       "role": "SOME_STRING"  
5669                     },  
5670                     "resources": [  
5671                       {  
5672                         "href": "/light",  
5673                         "rt": ["oic.r.light"],  
5674                         "if": ["oic.if.baseline", "oic.if.a"]  
5675                       },  
5676                       {  
5677                         "href": "/door",  
5678                         "rt": ["oic.r.door"],  
5679                         "if": ["oic.if.baseline", "oic.if.a"]  
5680                       }  
5681                     ]  
5682                   }  
5683                 ]  
5684               }  
5685           }  
5686         }  
5687       }  
5688     }  
5689   }  
5690 }
```

```

5680     }
5681     ],
5682     "permission": 24
5683   },
5684   {
5685     "aceid": 2,
5686     "subject": {
5687       "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5688     },
5689     "resources": [
5690       {
5691         "href": "/light",
5692         "rt": ["oic.r.light"],
5693         "if": ["oic.if.baseline", "oic.if.a"]
5694       },
5695       {
5696         "href": "/door",
5697         "rt": ["oic.r.door"],
5698         "if": ["oic.if.baseline", "oic.if.a"]
5699       }
5700     ],
5701     "permission": 24
5702   },
5703   {
5704     "aceid": 3,
5705     "subject": {"conntype": "anon-clear"},
5706     "resources": [
5707       {
5708         "href": "/light",
5709         "rt": ["oic.r.light"],
5710         "if": ["oic.if.baseline", "oic.if.a"]
5711       },
5712       {
5713         "href": "/door",
5714         "rt": ["oic.r.door"],
5715         "if": ["oic.if.baseline", "oic.if.a"]
5716       }
5717     ],
5718     "permission": 16,
5719     "validity": [
5720       {
5721         "period": "20160101T180000Z/20170102T070000Z",
5722         "recurrence": [ "DSTART:XXXXX",
5723           "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5724       },
5725       {
5726         "period": "20160101T180000Z/PT5H30M",
5727         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5728       }
5729     ]
5730   },
5731   ],
5732   "rowmeruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5733 },
5734 "schema": { "$ref": "#/definitions/Acl2" }
5735 },
5736 "400": {
5737   "description": "The request is invalid."
5738 }
5739 }
5740 },
5741 "post": {
5742   "description": "Updates the ACL resource with the provided ACEs.\n\nACES provided in the
5743 update with aceids not currently in the ACL\nresource are added.\n\nACES provided in the update with
5744 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL resource.\n\nACES provided in
5745 the update without aceid properties are added and\nassigned unique aceids in the ACL resource.\n",
5746   "parameters": [
5747     { "$ref": "#/parameters/interface" },
5748     { "$ref": "#/parameters/ace-filtered" },
5749   ],
5750   "name": "body",

```

```

5751     "in": "body",
5752     "required": true,
5753     "schema": { "$ref": "#/definitions/Acl2-Update" },
5754     "x-example":
5755     {
5756         "acllist2": [
5757             {
5758                 "aceid": 1,
5759                 "subject": {
5760                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5761                     "role": "SOME_STRING"
5762                 },
5763                 "resources": [
5764                     {
5765                         "href": "/light",
5766                         "xt": ["oic.r.light"],
5767                         "if": ["oic.if.baseline", "oic.if.a"]
5768                     },
5769                     {
5770                         "href": "/door",
5771                         "xt": ["oic.r.door"],
5772                         "if": ["oic.if.baseline", "oic.if.a"]
5773                     }
5774                 ],
5775                 "permission": 24
5776             },
5777             {
5778                 "aceid": 3,
5779                 "subject": {
5780                     "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5781                 },
5782                 "resources": [
5783                     {
5784                         "href": "/light",
5785                         "xt": ["oic.r.light"],
5786                         "if": ["oic.if.baseline", "oic.if.a"]
5787                     },
5788                     {
5789                         "href": "/door",
5790                         "xt": ["oic.r.door"],
5791                         "if": ["oic.if.baseline", "oic.if.a"]
5792                     }
5793                 ],
5794                 "permission": 24
5795             }
5796         ],
5797         "rowneruid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5798     }
5799 }
5800 },
5801 "responses": {
5802     "400": {
5803         "description": "The request is invalid."
5804     },
5805     "201": {
5806         "description": "The ACL entry is created."
5807     },
5808     "204": {
5809         "description": "The ACL entry is updated."
5810     }
5811 }
5812 },
5813 "delete": {
5814     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5815 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
5816 the\nspecified parameter are deleted.\n",
5817     "parameters": [
5818         {"$ref": "#/parameters/interface"},
5819         {"$ref": "#/parameters/ace-filtered"}
5820     ],
5821     "responses": {

```

```

5822         "200": {
5823             "description": "The matching ACEs or the entire ACL resource has been successfully
5824 deleted."
5825         },
5826         "400": {
5827             "description": "The request is invalid."
5828         }
5829     }
5830 }
5831 },
5832 },
5833 "parameters": {
5834     "interface": {
5835         "in": "query",
5836         "name": "if",
5837         "type": "string",
5838         "enum": ["oic.if.baseline"]
5839     },
5840     "ace-filtered": {
5841         "in": "query",
5842         "name": "aceid",
5843         "required": false,
5844         "type": "integer",
5845         "description": "Only applies to the ACE with the specified aceid",
5846         "x-example": 2112
5847     }
5848 },
5849 "definitions": {
5850     "Acl2": {
5851         "properties": {
5852             "owneruid": {
5853                 "description": "The value identifies the unique resource owner\nFormat pattern according
5854 to IETF RFC 4122.",
5855                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
5856 9]{12}$",
5857                 "type": "string"
5858             },
5859             "rt": {
5860                 "description": "Resource Type of the Resource",
5861                 "items": {
5862                     "maxLength": 64,
5863                     "type": "string",
5864                     "enum": ["oic.r.acl2"]
5865                 },
5866                 "minItems": 1,
5867                 "maxItems": 1,
5868                 "readOnly": true,
5869                 "type": "array"
5870             },
5871             "aclist2": {
5872                 "description": "Access Control Entries in the ACL resource",
5873                 "items": {
5874                     "properties": {
5875                         "aceid": {
5876                             "description": "An identifier for the ACE that is unique within the ACL. In cases
5877 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5878                             "minimum": 1,
5879                             "type": "integer"
5880                         },
5881                         "permission": {
5882                             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5883 permissions",
5884                             "x-detail-desc": [
5885                                 "0 - No permissions",
5886                                 "1 - Create permission is granted",
5887                                 "2 - Read, observe, discover permission is granted",
5888                                 "4 - Write, update permission is granted",
5889                                 "8 - Delete permission is granted",
5890                                 "16 - Notify permission is granted"
5891                             ],
5892                             "maximum": 31,

```

```

5893         "minimum": 0,
5894         "type": "integer"
5895     },
5896     "resources": {
5897         "description": "References the application's resources to which a security policy
5898 applies",
5899         "items": {
5900             "description": "Each resource must have at least one of these properties set",
5901             "properties": {
5902                 "href": {
5903                     "description": "When present, the ACE only applies when the href matches\nThis
5904 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5905                     "format": "uri",
5906                     "maxLength": 256,
5907                     "type": "string"
5908                 },
5909                 "if": {
5910                     "description": "When present, the ACE only applies when the if (interface)
5911 matches\nThe interface set supported by this resource",
5912                     "items": {
5913                         "enum": [
5914                             "oic.if.baseline",
5915                             "oic.if.ll",
5916                             "oic.if.b",
5917                             "oic.if.rw",
5918                             "oic.if.r",
5919                             "oic.if.a",
5920                             "oic.if.s"
5921                         ],
5922                         "type": "string"
5923                     },
5924                     "minItems": 1,
5925                     "type": "array"
5926                 },
5927                 "rt": {
5928                     "description": "When present, the ACE only applies when the rt (resource type)
5929 matches\nResource Type of the Resource",
5930                     "items": {
5931                         "maxLength": 64,
5932                         "type": "string"
5933                     },
5934                     "minItems": 1,
5935                     "type": "array"
5936                 },
5937                 "wc": {
5938                     "description": "A wildcard matching policy",
5939                     "pattern": "^[+*]$",
5940                     "type": "string"
5941                 }
5942             },
5943             "type": "object"
5944         },
5945         "type": "array"
5946     },
5947     "subject": {
5948         "anyOf": [
5949             {
5950                 "description": "Device identifier",
5951                 "properties": {
5952                     "uuid": {
5953                         "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5954 4122.",
5955                         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5956 fA-F0-9]{12}$",
5957                         "type": "string"
5958                     }
5959                 },
5960                 "required": [
5961                     "uuid"
5962                 ],
5963                 "type": "object"

```

```

5964     },
5965     {
5966         "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5967 <Authority> refers to the local entity or device.",
5968         "properties": {
5969             "authority": {
5970                 "description": "The Authority component of the entity being identified. A
5971 NULL <Authority> refers to the local entity or device.",
5972                 "type": "string"
5973             },
5974             "role": {
5975                 "description": "The ID of the role being identified.",
5976                 "type": "string"
5977             }
5978         },
5979         "required": [
5980             "role"
5981         ],
5982         "type": "object"
5983     },
5984     {
5985         "properties": {
5986             "conntype": {
5987                 "description": "This property allows an ACE to be matched based on the
5988 connection or message type",
5989                 "x-detail-desc": [
5990                     "auth-crypt - ACE applies if the Client is authenticated and the data
5991 channel or message is encrypted and integrity protected",
5992                     "anon-clear - ACE applies if the Client is not authenticated and the data
5993 channel or message is not encrypted but may be integrity protected"
5994                 ],
5995                 "enum": [
5996                     "auth-crypt",
5997                     "anon-clear"
5998                 ],
5999                 "type": "string"
6000             }
6001         },
6002         "required": [
6003             "conntype"
6004         ],
6005         "type": "object"
6006     }
6007 ]
6008 },
6009 "validity": {
6010     "description": "validity is an array of time-pattern objects",
6011     "items": {
6012         "description": "The time-pattern contains a period and recurrence expressed in
6013 IETF RFC 5545 syntax",
6014         "properties": {
6015             "period": {
6016                 "description": "String represents a period using the RFC5545 Period",
6017                 "type": "string"
6018             },
6019             "recurrence": {
6020                 "description": "String array represents a recurrence rule using the IETF RFC
6021 5545 Recurrence",
6022                 "items": {
6023                     "type": "string"
6024                 },
6025                 "type": "array"
6026             }
6027         },
6028         "required": [
6029             "period"
6030         ],
6031         "type": "object"
6032     },
6033     "type": "array"
6034 }

```

```

6035     },
6036     "required": [
6037         "aceid",
6038         "resources",
6039         "permission",
6040         "subject"
6041     ],
6042     "type": "object"
6043 },
6044 "type": "array"
6045 },
6046 "n": {
6047     "$ref":
6048 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6049 schema.json#/definitions/n"
6050 },
6051 "id": {
6052     "$ref":
6053 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6054 schema.json#/definitions/id"
6055 },
6056 "if" : {
6057     "description": "The interface set supported by this resource",
6058     "items": {
6059         "enum": [
6060             "oic.if.baseline"
6061         ],
6062         "type": "string"
6063     },
6064     "minItems": 1,
6065     "maxItems": 1,
6066     "readOnly": true,
6067     "type": "array"
6068 }
6069 },
6070 "type" : "object",
6071 "required": ["aclist2", "rowneruuid"]
6072 },
6073 "Acl2-Update" : {
6074     "properties": {
6075         "rowneruuid" : {
6076             "description": "The value identifies the unique resource owner\n Format pattern according
6077 to IETF RFC 4122.",
6078             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6079 9]{12}$",
6080             "type": "string"
6081         },
6082         "aclist2" : {
6083             "description": "Access Control Entries in the ACL resource",
6084             "items": {
6085                 "properties": {
6086                     "aceid": {
6087                         "description": "An identifier for the ACE that is unique within the ACL. In cases
6088 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
6089                         "minimum": 1,
6090                         "type": "integer"
6091                     },
6092                     "permission": {
6093                         "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
6094 permissions",
6095                         "x-detail-desc": [
6096                             "0 - No permissions",
6097                             "1 - Create permission is granted",
6098                             "2 - Read, observe, discover permission is granted",
6099                             "4 - Write, update permission is granted",
6100                             "8 - Delete permission is granted",
6101                             "16 - Notify permission is granted"
6102                         ],
6103                         "maximum": 31,
6104                         "minimum": 0,
6105                         "type": "integer"

```

```

6106     },
6107     "resources": {
6108         "description": "References the application's resources to which a security policy
6109 applies",
6110         "items": {
6111             "description": "Each resource must have at least one of these properties set",
6112             "properties": {
6113                 "href": {
6114                     "description": "When present, the ACE only applies when the href matches\nThis
6115 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6116                     "format": "uri",
6117                     "maxLength": 256,
6118                     "type": "string"
6119                 },
6120                 "if": {
6121                     "description": "When present, the ACE only applies when the if (interface)
6122 matches\nThe interface set supported by this resource",
6123                     "items": {
6124                         "enum": [
6125                             "oic.if.baseline",
6126                             "oic.if.ll",
6127                             "oic.if.b",
6128                             "oic.if.rw",
6129                             "oic.if.r",
6130                             "oic.if.a",
6131                             "oic.if.s"
6132                         ],
6133                         "type": "string"
6134                     },
6135                     "minItems": 1,
6136                     "type": "array"
6137                 },
6138                 "rt": {
6139                     "description": "When present, the ACE only applies when the rt (resource type)
6140 matches\nResource Type of the Resource",
6141                     "items": {
6142                         "maxLength": 64,
6143                         "type": "string"
6144                     },
6145                     "minItems": 1,
6146                     "type": "array"
6147                 },
6148                 "wc": {
6149                     "description": "A wildcard matching policy",
6150                     "x-detail-desc": [
6151                         "+ - Matches all discoverable resources",
6152                         "- - Matches all non-discoverable resources",
6153                         "* - Matches all resources"
6154                     ],
6155                     "enum": [
6156                         "+",
6157                         "-",
6158                         "*"
6159                     ],
6160                     "type": "string"
6161                 }
6162             },
6163             "type": "object"
6164         },
6165         "type": "array"
6166     },
6167     "subject": {
6168         "anyOf": [
6169             {
6170                 "description": "Device identifier",
6171                 "properties": {
6172                     "uuid": {
6173                         "description": "A UUID Device ID\n Format pattern according to IETF RFC
6174 4122.",
6175                         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
6176 fA-F0-9]{12}$",

```

```

6177         "type": "string"
6178     },
6179 },
6180 "required": [
6181     "uuid"
6182 ],
6183 "type": "object"
6184 },
6185 {
6186     "description": "Security role specified as an <Authority> & <Rolename>. A NULL
6187 <Authority> refers to the local entity or device.",
6188     "properties": {
6189         "authority": {
6190             "description": "The Authority component of the entity being identified. A
6191 NULL <Authority> refers to the local entity or device.",
6192             "type": "string"
6193         },
6194         "role": {
6195             "description": "The ID of the role being identified.",
6196             "type": "string"
6197         }
6198     },
6199 "required": [
6200     "role"
6201 ],
6202 "type": "object"
6203 },
6204 {
6205     "properties": {
6206         "conntype": {
6207             "description": "This property allows an ACE to be matched based on the
6208 connection or message type",
6209             "x-detail-desc": [
6210                 "auth-crypt - ACE applies if the Client is authenticated and the data
6211 channel or message is encrypted and integrity protected",
6212                 "anon-clear - ACE applies if the Client is not authenticated and the data
6213 channel or message is not encrypted but may be integrity protected"
6214             ],
6215             "enum": [
6216                 "auth-crypt",
6217                 "anon-clear"
6218             ],
6219             "type": "string"
6220         }
6221     },
6222 "required": [
6223     "conntype"
6224 ],
6225 "type": "object"
6226 }
6227 ]
6228 },
6229 "validity": {
6230     "description": "validity is an array of time-pattern objects",
6231     "items": {
6232         "description": "The time-pattern contains a period and recurrence expressed in
6233 IETF RFC 5545 syntax",
6234         "properties": {
6235             "period": {
6236                 "description": "String represents a period using the RFC5545 Period",
6237                 "type": "string"
6238             },
6239             "recurrence": {
6240                 "description": "String array represents a recurrence rule using the IETF RFC
6241 5545 Recurrence",
6242                 "items": {
6243                     "type": "string"
6244                 },
6245                 "type": "array"
6246             }
6247         }
6248     }
6249 }

```

```

6248         "required": [
6249             "period"
6250         ],
6251         "type": "object"
6252     },
6253     "type": "array"
6254 },
6255 },
6256     "required": [
6257         "resources",
6258         "permission",
6259         "subject"
6260     ],
6261     "type": "object"
6262 },
6263     "type": "array"
6264 },
6265 },
6266     "type": "object"
6267 }
6268 }
6269 }
6270

```

#### 6271 C.4.5 Property definition

6272 Table C.6 defines the Properties that are part of the ['oic.r.acl2'] Resource Type

6273 **Table C.4 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl2']**

Property name	Value type	Mandatory	Access mode	Description
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL resource
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruuid	string	Yes	Read Write	The value identifies the unique resource owner Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
aclist2	array: see schema	No	Read Write	Access Control Entries in the ACL resource
rowneruuid	string	No	Read Write	The value identifies the unique resource owner Format pattern according to IETF RFC 4122.

6274 **C.4.6 CRUDN behaviour**

6275 Table C.7 defines the CRUDN operations that are supported on the ['oic.r.acl2'] Resource Type

6276 **Table C.5 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl2']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

6277 **C.5 Managed Access Control**

6278 **C.5.1 Introduction**

6279 This resource specifies the host resources with access permission that is managed by an AMS.

6280

6281 **C.5.2 Well-known URI**

6282 /oic/sec/amacl

6283 **C.5.3 Resource type**

6284 The resource type (rt) is defined as: ['oic.r.amacl'].

6285 **C.5.4 OpenAPI 2.0 definition**

```

6286 {
6287   "swagger": "2.0",
6288   "info": {
6289     "title": "Managed Access Control",
6290     "version": "20190111",
6291     "license": {
6292       "name": "OCF Data Model License",
6293       "url":
6294         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6295         CENSE.md",
6296       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6297         reserved."
6298     },
6299     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6300   },
6301   "schemes": ["http"],
6302   "consumes": ["application/json"],
6303   "produces": ["application/json"],
6304   "paths": {
6305     "/oic/sec/amacl" : {
6306       "get": {
6307         "description": "This resource specifies the host resources with access permission that is
6308         managed by an AMS.\n",
6309         "parameters": [
6310           {"$ref": "#/parameters/interface"}
6311         ],
6312         "responses": {
6313           "200": {
6314             "description": "",
6315             "x-example":
6316               {
6317                 "rt" : ["oic.r.amacl"],
6318                 "resources": [
6319                   {
6320                     "href": "/temp",
6321                     "rt": ["oic.r.temperature"],
6322                     "if": ["oic.if.baseline", "oic.if.a"]
6323                   },
6324                   {
6325                     "href": "/temp",
6326                     "rt": ["oic.r.temperature"],
6327                     "if": ["oic.if.baseline", "oic.if.s"]
6328                   }
6329                 ]
6330               }
6331         }
6332       }
6333     }
6334   }

```

```

6331         "schema": { "$ref": "#/definitions/Amacl" }
6332     }
6333 },
6334 },
6335 "post": {
6336     "description": "Sets the new amacl data\n",
6337     "parameters": [
6338         { "$ref": "#/parameters/interface" },
6339         {
6340             "name": "body",
6341             "in": "body",
6342             "required": true,
6343             "schema": { "$ref": "#/definitions/Amacl" },
6344             "x-example":
6345                 {
6346                     "resources": [
6347                         {
6348                             "href": "/temp",
6349                             "rt": ["oic.r.temperature"],
6350                             "if": ["oic.if.baseline", "oic.if.a"]
6351                         },
6352                         {
6353                             "href": "/temp",
6354                             "rt": ["oic.r.temperature"],
6355                             "if": ["oic.if.baseline", "oic.if.s"]
6356                         }
6357                     ]
6358                 }
6359         ],
6360     },
6361     "responses": {
6362         "400": {
6363             "description": "The request is invalid."
6364         },
6365         "201": {
6366             "description": "The AMACL entry is created."
6367         },
6368         "204": {
6369             "description": "The AMACL entry is updated."
6370         }
6371     }
6372 },
6373 "put": {
6374     "description": "Creates the new acl data\n",
6375     "parameters": [
6376         { "$ref": "#/parameters/interface" },
6377         {
6378             "name": "body",
6379             "in": "body",
6380             "required": true,
6381             "schema": { "$ref": "#/definitions/Amacl" },
6382             "x-example":
6383                 {
6384                     "resources": [
6385                         {
6386                             "href": "/temp",
6387                             "rt": ["oic.r.temperature"],
6388                             "if": ["oic.if.baseline", "oic.if.a"]
6389                         },
6390                         {
6391                             "href": "/temp",
6392                             "rt": ["oic.r.temperature"],
6393                             "if": ["oic.if.baseline", "oic.if.s"]
6394                         }
6395                     ]
6396                 }
6397         ],
6398     },
6399     "responses": {
6400         "400": {
6401             "description": "The request is invalid."

```

```

6402     },
6403     "201": {
6404         "description": "The AMACL entry is created."
6405     }
6406 },
6407 },
6408 "delete": {
6409     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
6410 entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
6411 matched entry is deleted.\n",
6412     "parameters": [
6413         { "$ref": "#/parameters/interface"},
6414         {
6415             "in": "query",
6416             "description": "Delete the ACE identified by the string matching the subject value.\n",
6417             "type": "string",
6418             "name": "subject"
6419         }
6420     ],
6421     "responses": {
6422         "200": {
6423             "description": "The ACE instance or the the entire AMACL resource has been
6424 successfully deleted."
6425         },
6426         "400": {
6427             "description": "The request is invalid."
6428         }
6429     }
6430 },
6431 },
6432 },
6433 "parameters": {
6434     "interface": {
6435         "in": "query",
6436         "name": "if",
6437         "type": "string",
6438         "enum": ["oic.if.baseline"]
6439     }
6440 },
6441 "definitions": {
6442     "Amacl": {
6443         "properties": {
6444             "rt": {
6445                 "description": "Resource Type of the Resource",
6446                 "items": {
6447                     "maxLength": 64,
6448                     "type": "string",
6449                     "enum": ["oic.r.amacl"]
6450                 },
6451                 "minItems": 1,
6452                 "maxItems": 1,
6453                 "readOnly": true,
6454                 "type": "array"
6455             },
6456             "n": {
6457                 "$ref":
6458 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6459 schema.json#/definitions/n"
6460             },
6461             "id": {
6462                 "$ref":
6463 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6464 schema.json#/definitions/id"
6465             },
6466             "resources": {
6467                 "description": "Multiple links to this host's resources",
6468                 "items": {
6469                     "description": "Each resource must have at least one of these properties set",
6470                     "properties": {
6471                         "href": {
6472                             "description": "When present, the ACE only applies when the href matches\nThis is

```

```

6473 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6474     "format": "uri",
6475     "maxLength": 256,
6476     "type": "string"
6477   },
6478   "if": {
6479     "description": "When present, the ACE only applies when the if (interface)
6480 matches\nThe interface set supported by this resource",
6481     "items": {
6482       "enum": [
6483         "oic.if.baseline",
6484         "oic.if.ll",
6485         "oic.if.b",
6486         "oic.if.rw",
6487         "oic.if.r",
6488         "oic.if.a",
6489         "oic.if.s"
6490       ],
6491       "type": "string"
6492     },
6493     "minItems": 1,
6494     "type": "array"
6495   },
6496   "rt": {
6497     "description": "When present, the ACE only applies when the rt (resource type)
6498 matches\nResource Type of the Resource",
6499     "items": {
6500       "maxLength": 64,
6501       "type": "string"
6502     },
6503     "minItems": 1,
6504     "type": "array"
6505   },
6506   "wc": {
6507     "description": "A wildcard matching policy",
6508     "pattern": "^[~*]*$",
6509     "type": "string"
6510   }
6511 },
6512 "type": "object"
6513 },
6514 "type": "array"
6515 },
6516 "if": {
6517   "description": "The interface set supported by this resource",
6518   "items": {
6519     "enum": [
6520       "oic.if.baseline"
6521     ],
6522     "type": "string"
6523   },
6524   "minItems": 1,
6525   "maxItems": 1,
6526   "readOnly": true,
6527   "type": "array"
6528 }
6529 },
6530 "type": "object",
6531 "required": ["resources"]
6532 }
6533 }
6534 }
6535

```

### 6536 C.5.5 Property definition

6537 Table C.8 defines the Properties that are part of the ['oic.r.amacl'] Resource Type

6538 **Table C.6 – The Property definitions of the Resource with type 'rt' = ['oic.r.amacl']**

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

n	multiple types: see schema	No	Read Write	
resources	array: see schema	Yes	Read Write	Multiple links to this host's resources
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource

6539 **C.5.6 CRUDN behaviour**

6540 Table C.9 defines the CRUDN operations that are supported on the ['oic.r.amacl'] Resource Type

6541 **Table C.7 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.amacl']**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

6542 **C.6 Credential**

6543 **C.6.1 Introduction**

6544 This resource specifies credentials a device may use to establish secure communication.

6545 Retrieves the credential data.

6546 When used without query parameters, all the credential entries are returned.

6547 When used with a query parameter, only the credentials matching the specified  
6548 parameter are returned.

6549 Note that write-only credential data will not be returned.  
6550  
6551

6552 **C.6.2 Well-known URI**

6553 /oic/sec/cred

6554 **C.6.3 Resource type**

6555 The resource type (rt) is defined as: ['oic.r.cred'].

6556 **C.6.4 OpenAPI 2.0 definition**

```

6557 {
6558   "swagger": "2.0",
6559   "info": {
6560     "title": "Credential",
6561     "version": "v1.0-20181031",
6562     "license": {
6563       "name": "OCF Data Model License",
6564       "url":
6565         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6566         CENSE.md",
6567       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6568       reserved."
6569     },
6570     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6571   },
6572   "schemes": ["http"],
6573   "consumes": ["application/json"],
6574   "produces": ["application/json"],
6575   "paths": {
6576     "/oic/sec/cred" : {

```

```

6577     "get": {
6578         "description": "This resource specifies credentials a device may use to establish secure
6579 communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
6580 credential entries are returned.\nWhen used with a query parameter, only the credentials matching
6581 the specified\nparameter are returned.\n\nNote that write-only credential data will not be
6582 returned.\n",
6583         "parameters": [
6584             {"$ref": "#/parameters/interface"}
6585             , {"$ref": "#/parameters/cred-filtered-credid"}
6586             , {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6587         ],
6588         "responses": {
6589             "200": {
6590                 "description": "",
6591                 "x-example":
6592                 {
6593                     "rt": ["oic.r.cred"],
6594                     "creds": [
6595                         {
6596                             "credid": 55,
6597                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6598                             "roleid": {
6599                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6600                                 "role": "SOME_STRING"
6601                             },
6602                             "credtype": 32,
6603                             "publicdata": {
6604                                 "encoding": "oic.sec.encoding.base64",
6605                                 "data": "BASE-64-ENCODED-VALUE"
6606                             },
6607                             "privatedata": {
6608                                 "encoding": "oic.sec.encoding.base64",
6609                                 "data": "BASE-64-ENCODED-VALUE",
6610                                 "handle": 4
6611                             },
6612                             "optionaldata": {
6613                                 "revstat": false,
6614                                 "encoding": "oic.sec.encoding.base64",
6615                                 "data": "BASE-64-ENCODED-VALUE"
6616                             },
6617                             "period": "20160101T180000Z/20170102T070000Z",
6618                             "crms": [ "oic.sec.crm.pk10" ]
6619                         },
6620                         {
6621                             "credid": 56,
6622                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6623                             "roleid": {
6624                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6625                                 "role": "SOME_STRING"
6626                             },
6627                             "credtype": 1,
6628                             "publicdata": {
6629                                 "encoding": "oic.sec.encoding.base64",
6630                                 "data": "BASE-64-ENCODED-VALUE"
6631                             },
6632                             "privatedata": {
6633                                 "encoding": "oic.sec.encoding.base64",
6634                                 "data": "BASE-64-ENCODED-VALUE",
6635                                 "handle": 4
6636                             },
6637                             "optionaldata": {
6638                                 "revstat": false,
6639                                 "encoding": "oic.sec.encoding.base64",
6640                                 "data": "BASE-64-ENCODED-VALUE"
6641                             },
6642                             "period": "20160101T180000Z/20170102T070000Z",
6643                             "crms": [ "oic.sec.crm.pk10" ]
6644                         }
6645                     ],
6646                     "rowmeruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6647                 }
6648             }
6649         }

```

```

6648         ,
6649         "schema": { "$ref": "#/definitions/Cred" }
6650     },
6651     "400": {
6652         "description": "The request is invalid."
6653     }
6654 }
6655 },
6656 "post": {
6657     "description": "Updates the credential resource with the provided
6658 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
6659 resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
6660 resource completely replace the creds in the credential\nresource.\n\nCredentials provided in the
6661 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
6662 resource.\n",
6663     "parameters": [
6664         { "$ref": "#/parameters/interface" },
6665         {
6666             "name": "body",
6667             "in": "body",
6668             "required": true,
6669             "schema": { "$ref": "#/definitions/Cred-Update" },
6670             "x-example":
6671                 {
6672                     "creds": [
6673                         {
6674                             "credid": 55,
6675                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6676                             "roleid": {
6677                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6678                                 "role": "SOME_STRING"
6679                             },
6680                             "credtype": 32,
6681                             "publicdata": {
6682                                 "encoding": "oic.sec.encoding.base64",
6683                                 "data": "BASE-64-ENCODED-VALUE"
6684                             },
6685                             "privatedata": {
6686                                 "encoding": "oic.sec.encoding.base64",
6687                                 "data": "BASE-64-ENCODED-VALUE",
6688                                 "handle": 4
6689                             },
6690                             "optionaldata": {
6691                                 "revstat": false,
6692                                 "encoding": "oic.sec.encoding.base64",
6693                                 "data": "BASE-64-ENCODED-VALUE"
6694                             },
6695                             "period": "20160101T180000Z/20170102T070000Z",
6696                             "crms": [ "oic.sec.crm.pk10" ]
6697                         },
6698                         {
6699                             "credid": 56,
6700                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6701                             "roleid": {
6702                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6703                                 "role": "SOME_STRING"
6704                             },
6705                             "credtype": 1,
6706                             "publicdata": {
6707                                 "encoding": "oic.sec.encoding.base64",
6708                                 "data": "BASE-64-ENCODED-VALUE"
6709                             },
6710                             "privatedata": {
6711                                 "encoding": "oic.sec.encoding.base64",
6712                                 "data": "BASE-64-ENCODED-VALUE",
6713                                 "handle": 4
6714                             },
6715                             "optionaldata": {
6716                                 "revstat": false,
6717                                 "encoding": "oic.sec.encoding.base64",
6718                                 "data": "BASE-64-ENCODED-VALUE"

```

```

6719         },
6720         "period": "20160101T180000Z/20170102T070000Z",
6721         "crms": [ "oic.sec.crm.pk10" ]
6722     },
6723     ],
6724     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6725 }
6726 },
6727 ],
6728 "responses": {
6729     "400": {
6730         "description": "The request is invalid."
6731     },
6732     "201": {
6733         "description": "The credential entry is created."
6734     },
6735     "204": {
6736         "description": "The credential entry is updated."
6737     }
6738 },
6739 },
6740 "delete": {
6741     "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
6742 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
6743 matching\nthe query parameter are deleted.\n",
6744     "parameters": [
6745         {"$ref": "#/parameters/interface"},
6746         {"$ref": "#/parameters/cred-filtered-credid"},
6747         {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6748     ],
6749     "responses": {
6750         "400": {
6751             "description": "The request is invalid."
6752         },
6753         "204": {
6754             "description": "The specific credential(s) or the the entire credential resource has
6755 been successfully deleted."
6756         }
6757     }
6758 },
6759 },
6760 },
6761 "parameters": {
6762     "interface": {
6763         "in": "query",
6764         "name": "if",
6765         "type": "string",
6766         "enum": ["oic.if.baseline"]
6767     },
6768     "cred-filtered-credid": {
6769         "in": "query",
6770         "name": "credid",
6771         "required": false,
6772         "type": "integer",
6773         "description": "Only applies to the credential with the specified credid",
6774         "x-example": 2112
6775     },
6776     "cred-filtered-subjectuuid": {
6777         "in": "query",
6778         "name": "subjectuuid",
6779         "required": false,
6780         "type": "string",
6781         "description": "Only applies to credentials with the specified subject UUID",
6782         "x-example": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6783     }
6784 },
6785 "definitions": {
6786     "Cred": {
6787         "properties": {
6788             "rowneruuid": {
6789                 "description": "Format pattern according to IETF RFC 4122.",

```

```

6790     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6791 9]{12}$",
6792     "type": "string"
6793   },
6794   "rt" : {
6795     "description": "Resource Type of the Resource",
6796     "items": {
6797       "maxLength": 64,
6798       "type": "string",
6799       "enum": ["oic.r.cred"]
6800     },
6801     "minItems": 1,
6802     "readOnly": true,
6803     "type": "array",
6804     "uniqueItems": true
6805   },
6806   "n": {
6807     "$ref":
6808     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6809     schema.json#/definitions/n"
6810   },
6811   "id": {
6812     "$ref":
6813     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6814     schema.json#/definitions/id"
6815   },
6816   "creds" : {
6817     "description": "List of credentials available at this resource",
6818     "items": {
6819       "properties": {
6820         "credid": {
6821           "description": "Local reference to a credential resource",
6822           "type": "integer"
6823         },
6824         "credusage": {
6825           "description": "Representation of this credential's type\nCredential Types - Cred
6826 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6827 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
6828 password32 - Asymmetric encryption key",
6829           "maximum": 63,
6830           "minimum": 0,
6831           "type": "integer"
6832         },
6833         "credusage": {
6834           "description": "A string that provides hints about how/where the cred is used\nThe
6835 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6836 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6837 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
6838           "enum": [
6839             "oic.sec.cred.trustca",
6840             "oic.sec.cred.cert",
6841             "oic.sec.cred.rolecert",
6842             "oic.sec.cred.mfgtrustca",
6843             "oic.sec.cred.mfgcert"
6844           ],
6845           "type": "string"
6846         },
6847         "crms": {
6848           "description": "The refresh methods that may be used to update this credential",
6849           "items": {
6850             "description": "Each enum represents a method by which the credentials are
6851 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6852 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6853 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6854 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
6855             "enum": [
6856               "oic.sec.crm.pro",
6857               "oic.sec.crm.psk",
6858               "oic.sec.crm.rdp",
6859               "oic.sec.crm.skdc",
6860               "oic.sec.crm.pk10"

```

```

6861         },
6862         "type": "string"
6863     },
6864     "type": "array",
6865     "uniqueItems": true
6866 },
6867 "optionaldata": {
6868     "description": "Credential revocation status information\nOptional credential
6869 contents describes revocation status for this credential",
6870     "properties": {
6871         "data": {
6872             "description": "The encoded structure",
6873             "type": "string"
6874         },
6875         "encoding": {
6876             "description": "A string specifying the encoding format of the data contained in
6877 the optdata",
6878             "x-detail-desc": [
6879                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
6880                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
6881                 "oic.sec.encoding.base64 - Base64 encoded object",
6882                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
6883                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
6884                 "oic.sec.encoding.raw - Raw hex encoded data"
6885             ],
6886             "enum": [
6887                 "oic.sec.encoding.jwt",
6888                 "oic.sec.encoding.cwt",
6889                 "oic.sec.encoding.base64",
6890                 "oic.sec.encoding.pem",
6891                 "oic.sec.encoding.der",
6892                 "oic.sec.encoding.raw"
6893             ],
6894             "type": "string"
6895         },
6896         "revstat": {
6897             "description": "Revocation status flag - true = revoked",
6898             "type": "boolean"
6899         }
6900     },
6901     "required": [
6902         "revstat"
6903     ],
6904     "type": "object"
6905 },
6906 "period": {
6907     "description": "String with IETF RFC 5545 Period",
6908     "type": "string"
6909 },
6910 "privatedata": {
6911     "description": "Private credential information\nCredential resource non-public
6912 contents",
6913     "properties": {
6914         "data": {
6915             "description": "The encoded value",
6916             "maxLength": 3072,
6917             "type": "string"
6918         },
6919         "encoding": {
6920             "description": "A string specifying the encoding format of the data contained in
6921 the privdata\noic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT)
6922 encoding\noic.sec.encoding.cwt - IETF RFC 8392 encoding\noic.sec.encoding.base64 - Base64 encoded
6923 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6924 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data",
6925             "enum": [
6926                 "oic.sec.encoding.jwt",
6927                 "oic.sec.encoding.cwt",
6928                 "oic.sec.encoding.base64",
6929                 "oic.sec.encoding.uri",
6930                 "oic.sec.encoding.handle",
6931                 "oic.sec.encoding.raw"

```

```

6932         },
6933         "type": "string"
6934     },
6935     "handle": {
6936         "description": "Handle to a key storage resource",
6937         "type": "integer"
6938     }
6939 },
6940 "required": [
6941     "encoding"
6942 ],
6943 "type": "object"
6944 },
6945 "publicdata": {
6946     "description": "Public credential information",
6947     "properties": {
6948         "data": {
6949             "description": "The encoded value",
6950             "maxLength": 3072,
6951             "type": "string"
6952         },
6953         "encoding": {
6954             "description": "A string specifying the encoding format of the data contained in
6955 the pubdata\noic.sec.encoding.jwt - IETF RFC7519 JSON web token (JWT) encoding\noic.sec.encoding.cwt
6956 - IETF RFC 8392 encoding\noic.sec.encoding.base64 - Base64 encoded object\noic.sec.encoding.uri -
6957 URI reference\noic.sec.encoding.pem - Encoding for PEM encoded certificate or
6958 chain\noic.sec.encoding.der - Encoding for DER encoded certificate\noic.sec.encoding.raw - Raw hex
6959 encoded data",
6960             "enum": [
6961                 "oic.sec.encoding.jwt",
6962                 "oic.sec.encoding.cwt",
6963                 "oic.sec.encoding.base64",
6964                 "oic.sec.encoding.uri",
6965                 "oic.sec.encoding.pem",
6966                 "oic.sec.encoding.der",
6967                 "oic.sec.encoding.raw"
6968             ],
6969             "type": "string"
6970         }
6971     },
6972     "type": "object"
6973 },
6974 "roleid": {
6975     "description": "The role this credential possesses\nSecurity role specified as an
6976 <Authority> & <RoleName>. A NULL <Authority> refers to the local entity or device.",
6977     "properties": {
6978         "authority": {
6979             "description": "The Authority component of the entity being identified. A NULL
6980 <Authority> refers to the local entity or device.",
6981             "type": "string"
6982         },
6983         "role": {
6984             "description": "The ID of the role being identified.",
6985             "type": "string"
6986         }
6987     },
6988     "required": [
6989         "role"
6990     ],
6991     "type": "object"
6992 },
6993 "subjectuuid": {
6994     "anyOf": [
6995         {
6996             "description": "The id of the device, which the cred entry applies to or \"*\n"
6997 for wildcard identity",
6998             "pattern": "^\\*$",
6999             "type": "string"
7000         },
7001         {
7002             "description": "Format pattern according to IETF RFC 4122.",

```

```

7003         "pattern": "[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7004 F0-9]{12}$",
7005         "type": "string"
7006     }
7007 }
7008 }
7009 },
7010 "type": "object"
7011 },
7012 "type": "array"
7013 },
7014 "if" : {
7015     "description": "The interface set supported by this resource",
7016     "items": {
7017         "enum": [
7018             "oic.if.baseline"
7019         ],
7020         "type": "string"
7021     },
7022     "minItems": 1,
7023     "readOnly": true,
7024     "type": "array"
7025 }
7026 },
7027 "type" : "object",
7028 "required": ["creds", "rowneruuid"]
7029 },
7030 "Cred-Update" : {
7031     "properties": {
7032         "rowneruuid" : {
7033             "description": "Format pattern according to IETF RFC 4122.",
7034             "pattern": "[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7035 9]{12}$",
7036             "type": "string"
7037         },
7038         "creds" : {
7039             "description": "List of credentials available at this resource",
7040             "items": {
7041                 "properties": {
7042                     "credid": {
7043                         "description": "Local reference to a credential resource",
7044                         "type": "integer"
7045                     },
7046                     "credtype": {
7047                         "description": "Representation of this credential's type\nCredential Types - Cred
7048 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7049 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
7050 password32 - Asymmetric encryption key",
7051                         "maximum": 63,
7052                         "minimum": 0,
7053                         "type": "integer"
7054                     },
7055                     "credusage": {
7056                         "description": "A string that provides hints about how/where the cred is used\nThe
7057 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7058 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7059 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
7060                         "enum": [
7061                             "oic.sec.cred.trustca",
7062                             "oic.sec.cred.cert",
7063                             "oic.sec.cred.rolecert",
7064                             "oic.sec.cred.mfgtrustca",
7065                             "oic.sec.cred.mfgcert"
7066                         ],
7067                         "type": "string"
7068                     },
7069                     "crms": {
7070                         "description": "The refresh methods that may be used to update this credential",
7071                         "items": {
7072                             "description": "Each enum represents a method by which the credentials are
7073 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -

```

```

7074 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7075 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7076 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
7077     "enum": [
7078         "oic.sec.crm.pro",
7079         "oic.sec.crm.psk",
7080         "oic.sec.crm.rdp",
7081         "oic.sec.crm.skdc",
7082         "oic.sec.crm.pk10"
7083     ],
7084     "type": "string"
7085 },
7086     "type": "array"
7087 },
7088     "optionaldata": {
7089         "description": "Credential revocation status information\nOptional credential
7090 contents describes revocation status for this credential",
7091         "properties": {
7092             "data": {
7093                 "description": "The encoded structure",
7094                 "type": "string"
7095             },
7096             "encoding": {
7097                 "description": "A string specifying the encoding format of the data contained in
7098 the optdata",
7099                 "x-detail-desc": [
7100                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
7101                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7102                     "oic.sec.encoding.base64 - Base64 encoded object",
7103                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
7104                     "oic.sec.encoding.der - Encoding for DER encoded certificate",
7105                     "oic.sec.encoding.raw - Raw hex encoded data"
7106                 ],
7107                 "enum": [
7108                     "oic.sec.encoding.jwt",
7109                     "oic.sec.encoding.cwt",
7110                     "oic.sec.encoding.base64",
7111                     "oic.sec.encoding.pem",
7112                     "oic.sec.encoding.der",
7113                     "oic.sec.encoding.raw"
7114                 ],
7115                 "type": "string"
7116             },
7117             "revstat": {
7118                 "description": "Revocation status flag - true = revoked",
7119                 "type": "boolean"
7120             }
7121         },
7122         "required": [
7123             "revstat"
7124         ],
7125         "type": "object"
7126     },
7127     "period": {
7128         "description": "String with IETF RFC 5545 Period",
7129         "type": "string"
7130     },
7131     "privatedata": {
7132         "description": "Private credential information\nCredential resource non-public
7133 contents",
7134         "properties": {
7135             "data": {
7136                 "description": "The encoded value",
7137                 "maxLength": 3072,
7138                 "type": "string"
7139             },
7140             "encoding": {
7141                 "description": "A string specifying the encoding format of the data contained in
7142 the privdata",
7143                 "x-detail-desc": [
7144                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",

```

```

7145         "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7146         "oic.sec.encoding.base64 - Base64 encoded object",
7147         "oic.sec.encoding.uri - URI reference",
7148         "oic.sec.encoding.handle - Data is contained in a storage sub-system
7149 referenced using a handle",
7150         "oic.sec.encoding.raw - Raw hex encoded data"
7151     ],
7152     "enum": [
7153         "oic.sec.encoding.jwt",
7154         "oic.sec.encoding.cwt",
7155         "oic.sec.encoding.base64",
7156         "oic.sec.encoding.uri",
7157         "oic.sec.encoding.handle",
7158         "oic.sec.encoding.raw"
7159     ],
7160     "type": "string"
7161 },
7162 "handle": {
7163     "description": "Handle to a key storage resource",
7164     "type": "integer"
7165 },
7166 },
7167 "required": [
7168     "encoding"
7169 ],
7170 "type": "object"
7171 },
7172 "publicdata": {
7173     "properties": {
7174         "data": {
7175             "description": "The encoded value",
7176             "maxLength": 3072,
7177             "type": "string"
7178         },
7179         "encoding": {
7180             "description": "Public credential information\nA string specifying the encoding
7181 format of the data contained in the pubdata",
7182             "x-detail-desc": [
7183                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
7184                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7185                 "oic.sec.encoding.base64 - Base64 encoded object",
7186                 "oic.sec.encoding.uri - URI reference",
7187                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
7188                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
7189                 "oic.sec.encoding.raw - Raw hex encoded data"
7190             ],
7191             "enum": [
7192                 "oic.sec.encoding.jwt",
7193                 "oic.sec.encoding.cwt",
7194                 "oic.sec.encoding.base64",
7195                 "oic.sec.encoding.uri",
7196                 "oic.sec.encoding.pem",
7197                 "oic.sec.encoding.der",
7198                 "oic.sec.encoding.raw"
7199             ],
7200             "type": "string"
7201         }
7202     },
7203     "type": "object"
7204 },
7205 "roleid": {
7206     "description": "The role this credential possesses\nSecurity role specified as an
7207 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
7208     "properties": {
7209         "authority": {
7210             "description": "The Authority component of the entity being identified. A NULL
7211 <Authority> refers to the local entity or device.",
7212             "type": "string"
7213         },
7214         "role": {
7215             "description": "The ID of the role being identified.",

```

```

7216         "type": "string"
7217     },
7218     "required": [
7219         "role"
7220     ],
7221     "type": "object"
7222 },
7223 "subjectuuid": {
7224     "anyOf": [
7225         {
7226             "description": "The id of the device, which the cred entry applies to or \"*\"
7227 for wildcard identity",
7228             "pattern": "^\\*$",
7229             "type": "string"
7230         },
7231         {
7232             "description": "Format pattern according to IETF RFC 4122.",
7233             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7234 F0-9]{12}$",
7235             "type": "string"
7236         }
7237     ]
7238 },
7239     "type": "object"
7240 },
7241     "type": "array"
7242 },
7243     "type": "array"
7244 },
7245     "if" :
7246     {
7247         "description": "The interface set supported by this resource",
7248         "items": {
7249             "enum": [
7250                 "oic.if.baseline"
7251             ],
7252             "type": "string"
7253         },
7254         "minItems": 1,
7255         "readOnly": true,
7256         "type": "array"
7257     }
7258 },
7259     "type" : "object"
7260 }
7261 }
7262 }
7263

```

7264 **C.6.5 Property definition**

7265 Table C.10 defines the Properties that are part of the ['oic.r.cred'] Resource Type

7266 **Table C.8 – The Property definitions of the Resource with type 'rt' = ['oic.r.cred']**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	No	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
creds	array: see schema	No	Read Write	List of credentials available at this resource

creds	array: see schema	Yes	Read Write	List of credentials available at this resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
n	multiple types: see schema	No	Read Write	

### 7267 C.6.6 CRUDN behaviour

7268 Table C.11 defines the CRUDN operations that are supported on the ['oic.r.cred'] Resource Type

7269 **Table C.9 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.cred']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 7270 C.7 Certificate Revocation

### 7271 C.7.1 Introduction

7272 This resource specifies certificate revocation lists as X.509 objects.

### 7274 C.7.2 Well-known URI

7275 /oic/sec/crl

### 7276 C.7.3 Resource type

7277 The resource type (rt) is defined as: ['oic.r.crl'].

### 7278 C.7.4 OpenAPI 2.0 definition

```

7279 {
7280   "swagger": "2.0",
7281   "info": {
7282     "title": "Certificate Revocation",
7283     "version": "v1.0-20150819",
7284     "license": {
7285       "name": "OCF Data Model License",
7286       "url":
7287         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7288         CENSE.md",
7289       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7290         reserved."
7291     },
7292     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7293   },
7294   "schemes": ["http"],
7295   "consumes": ["application/json"],
7296   "produces": ["application/json"],
7297   "paths": {
7298     "/oic/sec/crl" : {
7299       "get": {
7300         "description": "This resource specifies certificate revocation lists as X.509 objects.\n",

```

```

7301     "parameters": [
7302       { "$ref": "#/parameters/interface" }
7303     ],
7304     "responses": {
7305       "200": {
7306         "description": "",
7307         "x-example":
7308           {
7309             "rt": ["oic.r.crl"],
7310             "crlid": 1,
7311             "thisupdate": "2016-04-12T23:20:50.52Z",
7312             "crldata": "Base64ENCODEDCRL"
7313           },
7314         "schema": { "$ref": "#/definitions/Crl" }
7315       }
7316     },
7317     "post": {
7318       "description": "Updates the CRL data\n",
7319       "parameters": [
7320         { "$ref": "#/parameters/interface" },
7321         {
7322           "name": "body",
7323           "in": "body",
7324           "required": true,
7325           "schema": { "$ref": "#/definitions/Crl-Update" },
7326           "x-example":
7327             {
7328               "crlid": 1,
7329               "thisupdate": "2016-04-12T23:20:50.52Z",
7330               "crldata": "Base64ENCODEDCRL"
7331             }
7332         }
7333       ],
7334       "responses": {
7335         "400": {
7336           "description": "The request is invalid."
7337         },
7338         "204": {
7339           "description": "The CRL entry is updated."
7340         }
7341       }
7342     }
7343   },
7344 },
7345 "parameters": {
7346   "interface": {
7347     "in": "query",
7348     "name": "if",
7349     "type": "string",
7350     "enum": ["oic.if.baseline"]
7351   }
7352 },
7353 "definitions": {
7354   "Crl": {
7355     "properties": {
7356       "rt": {
7357         "description": "Resource Type of the Resource",
7358         "items": {
7359           "maxLength": 64,
7360           "type": "string",
7361           "enum": ["oic.r.crl"]
7362         },
7363         "minItems": 1,
7364         "readOnly": true,
7365         "type": "array"
7366       },
7367       "n": {
7368         "$ref":
7369           "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-schema.json#/definitions/n"
7370       }
7371     }

```

```

7372     },
7373     "id": {
7374       "$ref":
7375       "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7376       schema.json#/definitions/id"
7377     },
7378     "crldata": {
7379       "description": "Base64 BER encoded crl data",
7380       "type": "string"
7381     },
7382     "crlid": {
7383       "description": "Local reference to a crl resource",
7384       "type": "integer"
7385     },
7386     "thisupdate": {
7387       "description": "UTC time of last CRL update",
7388       "type": "string"
7389     },
7390     "if": {
7391       "description": "The interface set supported by this resource",
7392       "items": {
7393         "enum": [
7394           "oic.if.baseline"
7395         ],
7396         "type": "string"
7397       },
7398       "minItems": 1,
7399       "readOnly": true,
7400       "type": "array"
7401     }
7402   },
7403   "type": "object",
7404   "required": ["crlid", "thisupdate", "crldata"]
7405 }
7406
7407 "Crl-Update": {
7408   "properties": {
7409     "crldata": {
7410       "description": "Base64 BER encoded crl data",
7411       "type": "string"
7412     },
7413     "crlid": {
7414       "description": "Local reference to a crl resource",
7415       "type": "integer"
7416     },
7417     "thisupdate": {
7418       "description": "UTC time of last CRL update",
7419       "type": "string"
7420     }
7421   },
7422   "type": "object"
7423 }
7424 }
7425 }
7426

```

### 7427 C.7.5 Property definition

7428 Table C.12 defines the Properties that are part of the ['oic.r.crl'] Resource Type

7429 **Table C.10 – The Property definitions of the Resource with type 'rt' = ['oic.r.crl']**

Property name	Value type	Mandatory	Access mode	Description
thisupdate	string	Yes	Read Write	UTC time of last CRL update
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	

if	array: schema	see	No	Read Only	The OCF Interface set supported by this resource
rt	array: schema	see	No	Read Only	Resource Type of the Resource
crldid	integer		Yes	Read Write	Local reference to a crl resource
crldata	string		Yes	Read Write	Base64 BER encoded crl data
thisupdate	string			Read Write	UTC time of last CRL update
crldid	integer			Read Write	Local reference to a crl resource
crldata	string			Read Write	Base64 BER encoded crl data

7430 **C.7.6 CRUDN behaviour**

7431 Table C.13 defines the CRUDN operations that are supported on the ['oic.r.crl'] Resource Type

7432 **Table C.11 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.crl']**

Create	Read	Update	Delete	Notify
	get	post		observe

7433 **C.8 Certificate Signing Request**

7434 **C.8.1 Introduction**

7435 This resource specifies a Certificate Signing Request.

7436

7437 **C.8.2 Well-known URI**

7438 /oic/sec/csr

7439 **C.8.3 Resource type**

7440 The resource type (rt) is defined as: ['oic.r.csr'].

7441 **C.8.4 OpenAPI 2.0 definition**

```

7442 {
7443   "swagger": "2.0",
7444   "info": {
7445     "title": "Certificate Signing Request",
7446     "version": "v1.0-20150819",
7447     "license": {
7448       "name": "OCF Data Model License",
7449       "url":
7450 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7451 CENSE.md",
7452     "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7453 reserved."
7454   },
7455   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7456 },
7457   "schemes": ["http"],
7458   "consumes": ["application/json"],
7459   "produces": ["application/json"],
7460   "paths": {
7461     "/oic/sec/csr" : {
7462       "get": {
7463         "description": "This resource specifies a Certificate Signing Request.\n",
7464         "parameters": [

```

```

7465     {"$ref": "#/parameters/interface"}
7466   },
7467   "responses": {
7468     "200": {
7469       "description": "",
7470       "x-example":
7471         {
7472           "rt": ["oic.r.csr"],
7473           "encoding": "oic.sec.encoding.pem",
7474           "csr": "PEMENCODEDCSR"
7475         },
7476       "schema": { "$ref": "#/definitions/Csr" }
7477     },
7478     "404": {
7479       "description": "The device does not support certificates and generating CSRs."
7480     },
7481     "503": {
7482       "description": "The device is not yet ready to return a response. Try again later."
7483     }
7484   }
7485 }
7486 },
7487 },
7488 "parameters": {
7489   "interface": {
7490     "in": "query",
7491     "name": "if",
7492     "type": "string",
7493     "enum": ["oic.if.baseline"]
7494   }
7495 },
7496 "definitions": {
7497   "Csr": {
7498     "properties": {
7499       "rt": {
7500         "description": "Resource Type of the Resource",
7501         "items": {
7502           "maxLength": 64,
7503           "type": "string",
7504           "enum": ["oic.r.csr"]
7505         },
7506         "minItems": 1,
7507         "readOnly": true,
7508         "type": "array"
7509       },
7510       "encoding": {
7511         "description": "A string specifying the encoding format of the data contained in csr",
7512         "x-detail-desc": [
7513           "oic.sec.encoding.pem - Encoding for PEM encoded CSR",
7514           "oic.sec.encoding.der - Encoding for DER encoded CSR"
7515         ],
7516         "enum": [
7517           "oic.sec.encoding.pem",
7518           "oic.sec.encoding.der"
7519         ],
7520         "readOnly": true,
7521         "type": "string"
7522       },
7523       "n": {
7524         "$ref":
7525           "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7526           schema.json#/definitions/n"
7527       },
7528       "id": {
7529         "$ref":
7530           "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7531           schema.json#/definitions/id"
7532       },
7533       "csr": {
7534         "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property",
7535         "maxLength": 3072,

```

```

7536     "readOnly": true,
7537     "type": "string"
7538   },
7539   "if": {
7540     "description": "The interface set supported by this resource",
7541     "items": {
7542       "enum": [
7543         "oic.if.baseline"
7544       ],
7545       "type": "string"
7546     },
7547     "minItems": 1,
7548     "readOnly": true,
7549     "type": "array"
7550   }
7551 },
7552 "type": "object",
7553 "required": ["csr", "encoding"]
7554 }
7555 }
7556 }
7557

```

### 7558 C.8.5 Property definition

7559 Table C.14 defines the Properties that are part of the ['oic.r.csr'] Resource Type

7560 **Table C.12 – The Property definitions of the Resource with type 'rt' = ['oic.r.csr']**

Property name	Value type	Mandatory	Access mode	Description
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in csr
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource

### 7561 C.8.6 CRUDN behaviour

7562 Table C.15 defines the CRUDN operations that are supported on the ['oic.r.csr'] Resource Type

7563 **Table C.13 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.csr']**

Create	Read	Update	Delete	Notify
	get			observe

## 7564 C.9 Device Owner Transfer Method

### 7565 C.9.1 Introduction

7566 This resource specifies properties needed to establish a device owner.

7567

### 7568 C.9.2 Well-known URI

7569 /oic/sec/doxm

### 7570 C.9.3 Resource type

7571 The resource type (rt) is defined as: ['oic.r.doxm'].

### 7572 C.9.4 OpenAPI 2.0 definition

```
7573 {
7574   "swagger": "2.0",
7575   "info": {
7576     "title": "Device Owner Transfer Method",
7577     "version": "v1.0-20181001",
7578     "license": {
7579       "name": "OCF Data Model License",
7580       "url":
7581         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7582 CENSE.md",
7583       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7584 reserved."
7585     },
7586     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7587   },
7588   "schemes": ["http"],
7589   "consumes": ["application/json"],
7590   "produces": ["application/json"],
7591   "paths": {
7592     "/oic/sec/doxm" : {
7593       "get": {
7594         "description": "This resource specifies properties needed to establish a device owner.\n",
7595         "parameters": [
7596           { "$ref": "#/parameters/interface" }
7597         ],
7598         "responses": {
7599           "200": {
7600             "description": "",
7601             "x-example":
7602               {
7603                 "rt": ["oic.r.doxm"],
7604                 "oxms": [ 0, 2, 3 ],
7605                 "oxmsel": 0,
7606                 "sct": 16,
7607                 "owned": true,
7608                 "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7609                 "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7610                 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7611               }
7612             ,
7613             "schema": { "$ref": "#/definitions/Doxm" }
7614           },
7615           "400": {
7616             "description": "The request is invalid."
7617           }
7618         }
7619       },
7620       "post": {
7621         "description": "Updates the DOXM resource data\n",
7622         "parameters": [
7623           { "$ref": "#/parameters/interface" },
7624           {
7625             "name": "body",
7626             "in": "body",
```

```

7627         "required": true,
7628         "schema": { "$ref": "#/definitions/Doxm-Update" },
7629         "x-example":
7630         {
7631             "oxmsel": 0,
7632             "owned": true,
7633             "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7634             "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7635             "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7636         }
7637     },
7638 },
7639     "responses": {
7640         "400": {
7641             "description": "The request is invalid."
7642         },
7643         "204": {
7644             "description": "The DOXM entry is updated."
7645         }
7646     }
7647 },
7648 },
7649 },
7650 "parameters": {
7651     "interface": {
7652         "in": "query",
7653         "name": "if",
7654         "type": "string",
7655         "enum": ["oic.if.baseline"]
7656     }
7657 },
7658 "definitions": {
7659     "Doxm": {
7660         "properties": {
7661             "rowneruuid": {
7662                 "description": "Format pattern according to IETF RFC 4122.",
7663                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7664                 "type": "string"
7665             },
7666             "oxms": {
7667                 "description": "List of supported owner transfer methods",
7668                 "items": {
7669                     "description": "The device owner transfer methods that may be selected at device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated)",
7670                     "type": "integer"
7671                 },
7672                 "readOnly": true,
7673                 "type": "array"
7674             },
7675             "devowneruuid": {
7676                 "description": "Format pattern according to IETF RFC 4122.",
7677                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7678                 "type": "string"
7679             },
7680             "deviceuuid": {
7681                 "description": "The uuid formatted identity of the device\nFormat pattern according to IETF RFC 4122.",
7682                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7683                 "type": "string"
7684             },
7685             "owned": {
7686                 "description": "Ownership status flag",
7687                 "type": "boolean"
7688             }
7689         }
7690     }
7691 },
7692 },
7693 },
7694 },
7695 },
7696 },
7697 }

```

```

7698     "n": {
7699       "$ref":
7700 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7701 schema.json#/definitions/n"
7702     },
7703     "id": {
7704       "$ref":
7705 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7706 schema.json#/definitions/id"
7707     },
7708     "oxmsel": {
7709       "description": "The selected owner transfer method used during on-boarding\nThe device
7710 owner transfer methods that may be selected at device on-boarding. Each value indicates a specific
7711 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7712 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7713 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7714 method (oic.sec.doxm.dcap) (deprecated)",
7715       "type": "integer"
7716     },
7717     "sct": {
7718       "description": "Bitmask encoding of supported credential types\nCredential Types -
7719 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7720 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7721 password32 - Asymmetric encryption key",
7722       "maximum": 63,
7723       "minimum": 0,
7724       "type": "integer",
7725       "readOnly": true
7726     },
7727     "rt" : {
7728       "description": "Resource Type of the Resource",
7729       "items": {
7730         "maxLength": 64,
7731         "type": "string",
7732         "enum": ["oic.r.doxm"]
7733       },
7734       "minItems": 1,
7735       "readOnly": true,
7736       "type": "array"
7737     },
7738     "if": {
7739       "description": "The interface set supported by this resource",
7740       "items": {
7741         "enum": [
7742           "oic.if.baseline"
7743         ],
7744         "type": "string"
7745       },
7746       "minItems": 1,
7747       "readOnly": true,
7748       "type": "array"
7749     },
7750     "type" : "object",
7751     "required": ["oxms", "oxmsel", "sct", "owned", "deviceuid", "devowneruid", "rowneruid"]
7752   },
7753   "Doxm-Update" : {
7754     "properties": {
7755       "rowneruid": {
7756         "description": "Format pattern according to IETF RFC 4122.",
7757         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7758 9]{12}$",
7759         "type": "string"
7760       },
7761       "devowneruid": {
7762         "description": "Format pattern according to IETF RFC 4122.",
7763         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7764 9]{12}$",
7765         "type": "string"
7766       },
7767       "deviceuid": {
7768

```

```

7769         "description": "The uuid formatted identity of the device\nFormat pattern according to
7770 IETF RFC 4122.",
7771         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7772 9]{12}$",
7773         "type": "string"
7774     },
7775     "owned": {
7776         "description": "Ownership status flag",
7777         "type": "boolean"
7778     },
7779     "oxmsel": {
7780         "description": "The selected owner transfer method used during on-boarding\nThe device
7781 owner transfer methods that may be selected at device on-boarding. Each value indicates a specific
7782 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7783 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7784 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7785 method (oic.sec.doxm.dcap) (deprecated)",
7786         "type": "integer"
7787     }
7788 },
7789 "type" : "object"
7790 }
7791 }
7792 }
7793

```

### 7794 C.9.5 Property definition

7795 Table C.16 defines the Properties that are part of the ['oic.r.doxm'] Resource Type

7796 **Table C.14 – The Property definitions of the Resource with type 'rt' = ['oic.r.doxm']**

Property name	Value type	Mandatory	Access mode	Description
devowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
oxms	array: see schema	Yes	Read Only	List of supported owner transfer methods
rt	array: see schema	No	Read Only	Resource Type of the Resource
owned	boolean	Yes	Read Write	Ownership status flag
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the device Format pattern according to IETF RFC 4122.
oxmsel	integer	Yes	Read Write	The selected owner transfer method used during on-boarding The device owner transfer methods that may be selected at device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier

				for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated)
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
n	multiple types: see schema	No	Read Write	
sct	integer	Yes	Read Only	Bitmask encoding of supported credential types Credential Types - Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 - Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or password32 - Asymmetric encryption key
id	multiple types: see schema	No	Read Write	
rowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string		Read Write	The uuid formatted identity of the device Format pattern according to IETF RFC 4122.
owned	boolean		Read Write	Ownership status flag
oxmsel	integer		Read Write	The selected owner transfer method used during on-boarding The device owner transfer methods that may be selected at device on-boarding. Each value indicates a specific Owner Transfer method0 - Numeric

				OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap) (deprecated)
devowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.

7797 **C.9.6 CRUDN behaviour**

7798 Table C.17 defines the CRUDN operations that are supported on the ['oic.r.doxm'] Resource Type

7799 **Table C.15 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.doxm']**

Create	Read	Update	Delete	Notify
	get	post		observe

7800 **C.10 Device Provisioning Status**

7801 **C.10.1 Introduction**

7802 This resource specifies device provisioning status.

7803

7804 **C.10.2 Well-known URI**

7805 /oic/sec/pstat

7806 **C.10.3 Resource type**

7807 The resource type (rt) is defined as: ['oic.r.pstat'].

7808 **C.10.4 OpenAPI 2.0 definition**

```
7809 {
7810   "swagger": "2.0",
7811   "info": {
7812     "title": "Device Provisioning Status",
7813     "version": "v1.0-20191001",
7814     "license": {
7815       "name": "OCF Data Model License",
7816       "url":
7817         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7818         CENSE.md",
7819       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7820       reserved."
7821     },
7822     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7823   },
7824   "schemes": ["http"],
7825   "consumes": ["application/json"],
7826   "produces": ["application/json"],
7827   "paths": {
7828     "/oic/sec/pstat" : {
```

```

7829 "get": {
7830   "description": "This resource specifies device provisioning status.\n",
7831   "parameters": [
7832     { "$ref": "#/parameters/interface" }
7833   ],
7834   "responses": {
7835     "200": {
7836       "description": "",
7837       "x-example":
7838         {
7839           "rt": ["oic.r.pstat"],
7840           "dos": {"s": 3, "p": true},
7841           "isop": true,
7842           "cm": 8,
7843           "tm": 60,
7844           "om": 2,
7845           "sm": 7,
7846           "rowneruid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7847         },
7848       "schema": { "$ref": "#/definitions/Pstat" }
7849     },
7850     "400": {
7851       "description": "The request is invalid."
7852     }
7853   }
7854 },
7855 "post": {
7856   "description": "Sets or updates device provisioning status data.\n",
7857   "parameters": [
7858     { "$ref": "#/parameters/interface" },
7859     {
7860       "name": "body",
7861       "in": "body",
7862       "required": true,
7863       "schema": { "$ref": "#/definitions/Pstat-Update" },
7864       "x-example":
7865         {
7866           "dos": {"s": 3},
7867           "tm": 60,
7868           "om": 2,
7869           "rowneruid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7870         }
7871     }
7872   ],
7873   "responses": {
7874     "400": {
7875       "description": "The request is invalid."
7876     },
7877     "204": {
7878       "description": "The PSTAT entry is updated."
7879     }
7880   }
7881 },
7882 },
7883 },
7884 "parameters": {
7885   "interface": {
7886     "in": "query",
7887     "name": "if",
7888     "type": "string",
7889     "enum": ["oic.if.baseline"]
7890   }
7891 },
7892 "definitions": {
7893   "Pstat": {
7894     "properties": {
7895       "rowneruid": {
7896         "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7897 according to IETF RFC 4122.",
7898         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7899 9]{12}$",

```

```

7900         "type": "string"
7901     },
7902     "rt": {
7903         "description": "Resource Type of the Resource",
7904         "items": {
7905             "maxLength": 64,
7906             "type": "string",
7907             "enum": ["oic.r.pstat"]
7908         },
7909         "minItems": 1,
7910         "readOnly": true,
7911         "type": "array"
7912     },
7913     "om": {
7914         "description": "Current operational mode\nDevice provisioning operation may be server
7915 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7916 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7917 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7918 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
7919         "maximum": 7,
7920         "minimum": 1,
7921         "type": "integer"
7922     },
7923     "cm": {
7924         "description": "Current device provisioning mode\nDevice provisioning mode maintains a
7925 bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
7926 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7927 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7928 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7929 Software Version Validation128 - Initiate Secure Software Update",
7930         "maximum": 255,
7931         "minimum": 0,
7932         "type": "integer",
7933         "readOnly": true
7934     },
7935     "n": {
7936         "$ref":
7937 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7938 schema.json#/definitions/n"
7939     },
7940     "id": {
7941         "$ref":
7942 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7943 schema.json#/definitions/id"
7944     },
7945     "isop": {
7946         "description": "true indicates device is operational",
7947         "readOnly": true,
7948         "type": "boolean"
7949     },
7950     "tm": {
7951         "description": "Target device provisioning mode\nDevice provisioning mode maintains a
7952 bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
7953 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7954 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7955 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7956 Software Version Validation128 - Initiate Secure Software Update",
7957         "maximum": 255,
7958         "minimum": 0,
7959         "type": "integer"
7960     },
7961     "sm": {
7962         "description": "Supported operational modes\nDevice provisioning operation may be server
7963 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7964 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7965 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7966 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
7967         "maximum": 7,
7968         "minimum": 1,
7969         "type": "integer",
7970         "readOnly": true

```

```

7971     },
7972     "dos": {
7973         "description": "Device on-boarding state\nDevice operation state machine",
7974         "properties": {
7975             "p": {
7976                 "default": true,
7977                 "description": "'p' is TRUE when the 's' state is pending until all necessary changes
to device resources are complete.",
7978                 "readOnly": true,
7979                 "type": "boolean"
7980             },
7981         },
7982         "s": {
7983             "description": "The current or pending operational state",
7984             "x-detail-desc": [
7985                 "0 - RESET - Device reset state",
7986                 "1 - RFOTM - Ready for device owner transfer method state",
7987                 "2 - RFPRO - Ready for device provisioning state",
7988                 "3 - RFNOP - Ready for device normal operation state",
7989                 "4 - SRESET - The device is in a soft reset state"
7990             ],
7991             "maximum": 4,
7992             "minimum": 0,
7993             "type": "integer"
7994         },
7995     },
7996     "required": [
7997         "s"
7998     ],
7999     "type": "object"
8000 },
8001 "if" : {
8002     "description": "The interface set supported by this resource",
8003     "items": {
8004         "enum": [
8005             "oic.if.baseline"
8006         ],
8007         "type": "string"
8008     },
8009     "minItems": 1,
8010     "readOnly": true,
8011     "type": "array"
8012 },
8013 },
8014 "type" : "object",
8015 "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
8016 },
8017 "Pstat-Update" : {
8018     "properties": {
8019         "rowneruuid": {
8020             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
according to IETF RFC 4122.",
8021             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
8022             9]{12}$",
8023             "type": "string"
8024         },
8025     },
8026     "om": {
8027         "description": "Current operational mode\nDevice provisioning operation may be server
directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
8028 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
8029 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
8030 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
8031         "maximum": 7,
8032         "minimum": 1,
8033         "type": "integer"
8034     },
8035 },
8036     "tm": {
8037         "description": "Target device provisioning mode\nDevice provisioning mode maintains a
bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
8038 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
8039 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
8040 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
8041

```

```

8042 Software Version Validation128 - Initiate Secure Software Update",
8043     "maximum": 255,
8044     "minimum": 0,
8045     "type": "integer"
8046   },
8047   "dos": {
8048     "description": "Device on-boarding state\nDevice operation state machine",
8049     "properties": {
8050       "p": {
8051         "default": true,
8052         "description": "'p' is TRUE when the 's' state is pending until all necessary changes
8053 to device resources are complete.",
8054         "readOnly": true,
8055         "type": "boolean"
8056       },
8057       "s": {
8058         "description": "The current or pending operational state",
8059         "x-detail-desc": [
8060           "0 - RESET - Device reset state",
8061           "1 - RFOTM - Ready for device owner transfer method state",
8062           "2 - RFPRO - Ready for device provisioning state",
8063           "3 - RFNOP - Ready for device normal operation state",
8064           "4 - SRESET - The device is in a soft reset state"
8065         ],
8066         "maximum": 4,
8067         "minimum": 0,
8068         "type": "integer"
8069       }
8070     },
8071     "required": [
8072       "s"
8073     ],
8074     "type": "object"
8075   }
8076 },
8077 "type" : "object"
8078 }
8079 }
8080 }
8081

```

### 8082 C.10.5 Property definition

8083 Table C.18 defines the Properties that are part of the ['oic.r.pstat'] Resource Type

8084 **Table C.16 – The Property definitions of the Resource with type 'rt' = ['oic.r.pstat']**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.
sm	integer	Yes	Read Only	Supported operational modes Device provisioning operation may be server directed

				or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused
rt	array: see schema	No	Read Only	Resource Type of the Resource
isop	boolean	Yes	Read Only	true indicates device is operational
tm	integer	Yes	Read Write	Target device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of

				credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update
n	multiple types: see schema	No	Read Write	
dos	object: see schema	Yes	Read Write	Device onboarding state Device operation state machine
id	multiple types: see schema	No	Read Write	
om	integer	Yes	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused
cm	integer	Yes	Read Only	Current device provisioning

				<p>mode</p> <p>Device provisioning mode maintains a bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 -  Manufacturer reset state2 -  Device pairing and owner transfer state4 -  Unused8 -  Provisioning of credential management services16 -  Provisioning of access management services32 -  Provisioning of local ACLs64 -  Initiate Software Version Validation128 -  Initiate Secure Software Update</p>
dos	object: see schema	No	Read Write	<p>Device on-boarding state2  Device operation state machine</p>
rowneruuid	string	No	Read Write	<p>The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.</p>
om	integer	No	Read Write	<p>Current operational mode  Device provisioning operation may be server directed</p>

				<p>or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes</p> <ul style="list-style-type: none"> <li>1 - Server-directed utilizing multiple provisioning services</li> <li>2 - Server-directed utilizing a single provisioning service</li> <li>4 - Client-directed provisioning</li> <li>8 - Unused</li> <li>16 - Unused</li> <li>32 - Unused</li> <li>64 - Unused</li> <li>128 - Unused</li> </ul>
tm	integer	No	Read Write	<p>Target device provisioning mode</p> <p>Device provisioning mode maintains a bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value</p> <ul style="list-style-type: none"> <li>1 - Manufacturer reset state</li> <li>2 - Device pairing and owner transfer state</li> <li>4 - Unused</li> <li>8 - Provisioning of credential management services</li> <li>16 - Provisioning of access</li> </ul>

				management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update
--	--	--	--	--

8085 **C.10.6 CRUDN behaviour**

8086 Table C.19 defines the CRUDN operations that are supported on the ['oic.r.pstat'] Resource Type

8087 **Table C.17 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.pstat']**

Create	Read	Update	Delete	Notify
	get	post		observe

8088 **C.11 Asserted Roles**

8089 **C.11.1 Introduction**

8090 This Resource specifies roles that have been asserted.

8091

8092 **C.11.2 Well-known URI**

8093 /oic/sec/roles

8094 **C.11.3 Resource type**

8095 The resource type (rt) is defined as: ['oic.r.roles'].

8096 **C.11.4 OpenAPI 2.0 definition**

```

8097 {
8098   "swagger": "2.0",
8099   "info": {
8100     "title": "Asserted Roles",
8101     "version": "v1.0-20170323",
8102     "license": {
8103       "name": "OCF Data Model License",
8104       "url":
8105         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8106         CENSE.md",
8107       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8108       reserved."
8109     },
8110     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8111   },
8112   "schemes": ["http"],
8113   "consumes": ["application/json"],
8114   "produces": ["application/json"],
8115   "paths": {
8116     "/oic/sec/roles": {
8117       "get": {
8118         "description": "This Resource specifies roles that have been asserted.\n",
8119         "parameters": [
8120           {"$ref": "#/parameters/interface"}
8121         ],
8122         "responses": {
8123           "200": {
8124             "description": "",
8125             "x-example":
8126               {
8127                 "roles": [
8128                   {
8129                     "credid": 1,

```

```

8130         "credtype":8,
8131         "subjectuuiid":"00000000-0000-0000-0000-000000000000",
8132         "publicdata":
8133         {
8134             "encoding":"oic.sec.encoding.pem",
8135             "data":"PEMENCODEDROLECERT"
8136         },
8137         "optionaldata":
8138         {
8139             "revstat": false,
8140             "encoding":"oic.sec.encoding.pem",
8141             "data":"PEMENCODEDISSUERCERT"
8142         }
8143     },
8144     {
8145         "credid":2,
8146         "credtype":8,
8147         "subjectuuiid":"00000000-0000-0000-0000-000000000000",
8148         "publicdata":
8149         {
8150             "encoding":"oic.sec.encoding.pem",
8151             "data":"PEMENCODEDROLECERT"
8152         },
8153         "optionaldata":
8154         {
8155             "revstat": false,
8156             "encoding":"oic.sec.encoding.pem",
8157             "data":"PEMENCODEDISSUERCERT"
8158         }
8159     }
8160 ],
8161 "rt":["oic.r.roles"],
8162 "if":["oic.if.baseline"]
8163 }
8164 ,
8165 "schema": { "$ref": "#/definitions/Roles" }
8166 },
8167 "400": {
8168     "description" : "The request is invalid."
8169 }
8170 },
8171 },
8172 "post": {
8173     "description": "Update the roles resource, i.e., assert new roles to this server.\n\nNew
8174 role certificates that match an existing certificate (i.e., publicdata\and optionaldata are the
8175 same) are not added to the resource (and 204 is\nreturned).\n\nThe provided credid values are
8176 ignored, the resource assigns its own.\n",
8177     "parameters": [
8178         { "$ref": "#/parameters/interface"},
8179         {
8180             "name": "body",
8181             "in": "body",
8182             "required": true,
8183             "schema": { "$ref": "#/definitions/Roles-update" },
8184             "x-example":
8185             {
8186                 "roles" :[
8187                 {
8188                     "credid":1,
8189                     "credtype":8,
8190                     "subjectuuiid":"00000000-0000-0000-0000-000000000000",
8191                     "publicdata":
8192                     {
8193                         "encoding":"oic.sec.encoding.pem",
8194                         "data":"PEMENCODEDROLECERT"
8195                     },
8196                     "optionaldata":
8197                     {
8198                         "revstat": false,
8199                         "encoding":"oic.sec.encoding.pem",
8200                         "data":"PEMENCODEDISSUERCERT"

```

```

8201     }
8202   },
8203   {
8204     "credid":2,
8205     "credtype":8,
8206     "subjectuid":"00000000-0000-0000-0000-000000000000",
8207     "publicdata":
8208     {
8209       "encoding":"oic.sec.encoding.pem",
8210       "data":"PEMENCODEDROLECERT"
8211     },
8212     "optionaldata":
8213     {
8214       "revstat": false,
8215       "encoding":"oic.sec.encoding.pem",
8216       "data":"PEMENCODEDISSUERCERT"
8217     }
8218   }
8219 ],
8220 "rt":["oic.r.roles"],
8221 "if":["oic.if.baseline"]
8222 }
8223 }
8224 ],
8225 "responses": {
8226   "400": {
8227     "description" : "The request is invalid."
8228   },
8229   "204": {
8230     "description" : "The roles entry is updated."
8231   }
8232 }
8233 },
8234 "delete": {
8235   "description": "Deletes roles resource entries.\nWhen DELETE is used without query
8236 parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
8237 entries matching\nthe query parameter are deleted.\n",
8238   "parameters": [
8239     {"$ref": "#/parameters/interface"},
8240     {"$ref": "#/parameters/roles-filtered"}
8241   ],
8242   "responses": {
8243     "200": {
8244       "description" : "The specified or all roles resource entries have been successfully
8245 deleted."
8246     },
8247     "400": {
8248       "description" : "The request is invalid."
8249     }
8250   }
8251 }
8252 },
8253 },
8254 "parameters": {
8255   "interface" : {
8256     "in" : "query",
8257     "name" : "if",
8258     "type" : "string",
8259     "enum" : ["oic.if.baseline"]
8260   },
8261   "roles-filtered" : {
8262     "in" : "query",
8263     "name" : "credid",
8264     "required" : false,
8265     "type" : "integer",
8266     "description" : "Only applies to the credential with the specified credid",
8267     "x-example" : 2112
8268   }
8269 },
8270 "definitions": {
8271   "Roles" : {

```

```

8272     "properties": {
8273       "rt": {
8274         "description": "Resource Type of the Resource",
8275         "items": {
8276           "maxLength": 64,
8277           "type": "string",
8278           "enum": ["oic.r.roles"]
8279         },
8280         "minItems": 1,
8281         "readOnly": true,
8282         "type": "array"
8283       },
8284       "n": {
8285         "$ref":
8286         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8287         schema.json#/definitions/n"
8288       },
8289       "id": {
8290         "$ref":
8291         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8292         schema.json#/definitions/id"
8293       },
8294       "roles": {
8295         "description": "List of role certificates",
8296         "items": {
8297           "properties": {
8298             "credid": {
8299               "description": "Local reference to a credential resource",
8300               "type": "integer"
8301             },
8302             "credtype": {
8303               "description": "Representation of this credential's type\nCredential Types - Cred
8304               type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8305               Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
8306               password32 - Asymmetric encryption key",
8307               "maximum": 63,
8308               "minimum": 0,
8309               "type": "integer"
8310             },
8311             "credusage": {
8312               "description": "A string that provides hints about how/where the cred is used\nThe
8313               type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8314               Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8315               Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
8316               "enum": [
8317                 "oic.sec.cred.trustca",
8318                 "oic.sec.cred.cert",
8319                 "oic.sec.cred.rolecert",
8320                 "oic.sec.cred.mfgtrustca",
8321                 "oic.sec.cred.mfgcert"
8322               ],
8323               "type": "string"
8324             },
8325             "crms": {
8326               "description": "The refresh methods that may be used to update this credential",
8327               "items": {
8328                 "description": "Each enum represents a method by which the credentials are
8329                 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8330                 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8331                 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8332                 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
8333                 "enum": [
8334                   "oic.sec.crm.pro",
8335                   "oic.sec.crm.psk",
8336                   "oic.sec.crm.rdp",
8337                   "oic.sec.crm.skdc",
8338                   "oic.sec.crm.pk10"
8339                 ],
8340                 "type": "string"
8341               },
8342               "type": "array"

```

```

8343     },
8344     "optionaldata": {
8345       "description": "Credential revocation status information\nOptional credential
8346 contents describes revocation status for this credential",
8347       "properties": {
8348         "data": {
8349           "description": "The encoded structure",
8350           "type": "string"
8351         },
8352         "encoding": {
8353           "description": "A string specifying the encoding format of the data contained in
8354 the optdata",
8355           "x-detail-desc": [
8356             "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8357             "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8358             "oic.sec.encoding.base64 - Base64 encoded object",
8359             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8360             "oic.sec.encoding.der - Encoding for DER encoded certificate",
8361             "oic.sec.encoding.raw - Raw hex encoded data"
8362           ],
8363           "enum": [
8364             "oic.sec.encoding.jwt",
8365             "oic.sec.encoding.cwt",
8366             "oic.sec.encoding.base64",
8367             "oic.sec.encoding.pem",
8368             "oic.sec.encoding.der",
8369             "oic.sec.encoding.raw"
8370           ],
8371           "type": "string"
8372         },
8373         "revstat": {
8374           "description": "Revocation status flag - true = revoked",
8375           "type": "boolean"
8376         }
8377       },
8378       "required": [
8379         "revstat"
8380       ],
8381       "type": "object"
8382     },
8383     "period": {
8384       "description": "String with IETF RFC 5545 Period",
8385       "type": "string"
8386     },
8387     "privatedata": {
8388       "description": "Private credential information\nCredential resource non-public
8389 contents",
8390       "properties": {
8391         "data": {
8392           "description": "The encoded value",
8393           "maxLength": 3072,
8394           "type": "string"
8395         },
8396         "encoding": {
8397           "description": "A string specifying the encoding format of the data contained in
8398 the privdata",
8399           "x-detail-desc": [
8400             "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8401             "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8402             "oic.sec.encoding.base64 - Base64 encoded object",
8403             "oic.sec.encoding.uri - URI reference",
8404             "oic.sec.encoding.handle - Data is contained in a storage sub-system
8405 referenced using a handle",
8406             "oic.sec.encoding.raw - Raw hex encoded data"
8407           ],
8408           "enum": [
8409             "oic.sec.encoding.jwt",
8410             "oic.sec.encoding.cwt",
8411             "oic.sec.encoding.base64",
8412             "oic.sec.encoding.uri",
8413             "oic.sec.encoding.handle",

```

```

8414         "oic.sec.encoding.raw"
8415     ],
8416     "type": "string"
8417 },
8418 "handle": {
8419     "description": "Handle to a key storage resource",
8420     "type": "integer"
8421 }
8422 },
8423 "required": [
8424     "encoding"
8425 ],
8426 "type": "object"
8427 },
8428 "publicdata": {
8429     "description": "Public credential information",
8430     "properties": {
8431         "data": {
8432             "description": "The encoded value",
8433             "maxLength": 3072,
8434             "type": "string"
8435         },
8436         "encoding": {
8437             "description": "A string specifying the encoding format of the data contained in
8438 the pubdata",
8439             "x-detail-desc": [
8440                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8441                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8442                 "oic.sec.encoding.base64 - Base64 encoded object",
8443                 "oic.sec.encoding.uri - URI reference",
8444                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8445                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
8446                 "oic.sec.encoding.raw - Raw hex encoded data"
8447             ],
8448             "enum": [
8449                 "oic.sec.encoding.jwt",
8450                 "oic.sec.encoding.cwt",
8451                 "oic.sec.encoding.base64",
8452                 "oic.sec.encoding.uri",
8453                 "oic.sec.encoding.pem",
8454                 "oic.sec.encoding.der",
8455                 "oic.sec.encoding.raw"
8456             ],
8457             "type": "string"
8458         }
8459     },
8460     "type": "object"
8461 },
8462 "roleid": {
8463     "description": "The role this credential possesses\nSecurity role specified as an
8464 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
8465     "properties": {
8466         "authority": {
8467             "description": "The Authority component of the entity being identified. A NULL
8468 <Authority> refers to the local entity or device.",
8469             "type": "string"
8470         },
8471         "role": {
8472             "description": "The ID of the role being identified.",
8473             "type": "string"
8474         }
8475     },
8476     "required": [
8477         "role"
8478     ],
8479     "type": "object"
8480 },
8481 "subjectuuid": {
8482     "anyOf": [
8483         {
8484             "description": "The id of the device, which the cred entry applies to or \"*\

```

```

8485 for wildcard identity",
8486     "pattern": "^\\*$",
8487     "type": "string"
8488   },
8489   {
8490     "description": "Format pattern according to IETF RFC 4122.",
8491     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8492 F0-9]{12}$",
8493     "type": "string"
8494   }
8495 ]
8496 },
8497 },
8498 "type": "object"
8499 },
8500 "type": "array"
8501 },
8502 "if": {
8503   "description": "The interface set supported by this resource",
8504   "items": {
8505     "enum": [
8506       "oic.if.baseline"
8507     ],
8508     "type": "string"
8509   },
8510   "minItems": 1,
8511   "readOnly": true,
8512   "type": "array"
8513 }
8514 },
8515 "type": "object",
8516 "required": ["roles"]
8517 },
8518 "Roles-update" : {
8519   "properties": {
8520     "roles": {
8521       "description": "List of role certificates",
8522       "items": {
8523         "properties": {
8524           "credid": {
8525             "description": "Local reference to a credential resource",
8526             "type": "integer"
8527           },
8528           "credtype": {
8529             "description": "Representation of this credential's type\nCredential Types - Cred
8530 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8531 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
8532 password32 - Asymmetric encryption key",
8533             "maximum": 63,
8534             "minimum": 0,
8535             "type": "integer"
8536           },
8537           "credusage": {
8538             "description": "A string that provides hints about how/where the cred is used\nThe
8539 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8540 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8541 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
8542             "enum": [
8543               "oic.sec.cred.trustca",
8544               "oic.sec.cred.cert",
8545               "oic.sec.cred.rolecert",
8546               "oic.sec.cred.mfgtrustca",
8547               "oic.sec.cred.mfgcert"
8548             ],
8549             "type": "string"
8550           },
8551           "crms": {
8552             "description": "The refresh methods that may be used to update this credential",
8553             "items": {
8554               "description": "Each enum represents a method by which the credentials are
8555 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -

```

```

8556 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8557 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8558 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
8559     "enum": [
8560         "oic.sec.crm.pro",
8561         "oic.sec.crm.psk",
8562         "oic.sec.crm.rdp",
8563         "oic.sec.crm.skdc",
8564         "oic.sec.crm.pk10"
8565     ],
8566     "type": "string"
8567 },
8568     "type": "array"
8569 },
8570     "optionaldata": {
8571         "description": "Credential revocation status information\nOptional credential
8572 contents describes revocation status for this credential",
8573         "properties": {
8574             "data": {
8575                 "description": "The encoded structure",
8576                 "type": "string"
8577             },
8578             "encoding": {
8579                 "description": "A string specifying the encoding format of the data contained in
8580 the optdata",
8581                 "x-detail-desc": [
8582                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8583                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8584                     "oic.sec.encoding.base64 - Base64 encoded object",
8585                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8586                     "oic.sec.encoding.der - Encoding for DER encoded certificate",
8587                     "oic.sec.encoding.raw - Raw hex encoded data"
8588                 ],
8589                 "enum": [
8590                     "oic.sec.encoding.jwt",
8591                     "oic.sec.encoding.cwt",
8592                     "oic.sec.encoding.base64",
8593                     "oic.sec.encoding.pem",
8594                     "oic.sec.encoding.der",
8595                     "oic.sec.encoding.raw"
8596                 ],
8597                 "type": "string"
8598             },
8599             "revstat": {
8600                 "description": "Revocation status flag - true = revoked",
8601                 "type": "boolean"
8602             }
8603         },
8604         "required": [
8605             "revstat"
8606         ],
8607         "type": "object"
8608     },
8609     "period": {
8610         "description": "String with IETF RFC 5545 Period",
8611         "type": "string"
8612     },
8613     "privatedata": {
8614         "description": "Private credential information\nCredential resource non-public
8615 contents",
8616         "properties": {
8617             "data": {
8618                 "description": "The encoded value",
8619                 "maxLength": 3072,
8620                 "type": "string"
8621             },
8622             "encoding": {
8623                 "description": "A string specifying the encoding format of the data contained in
8624 the privdata",
8625                 "x-detail-desc": [
8626                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",

```

```

8627         "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8628         "oic.sec.encoding.base64 - Base64 encoded object",
8629         "oic.sec.encoding.uri - URI reference",
8630         "oic.sec.encoding.handle - Data is contained in a storage sub-system
8631 referenced using a handle",
8632         "oic.sec.encoding.raw - Raw hex encoded data"
8633     ],
8634     "enum": [
8635         "oic.sec.encoding.jwt",
8636         "oic.sec.encoding.cwt",
8637         "oic.sec.encoding.base64",
8638         "oic.sec.encoding.uri",
8639         "oic.sec.encoding.handle",
8640         "oic.sec.encoding.raw"
8641     ],
8642     "type": "string"
8643 },
8644 "handle": {
8645     "description": "Handle to a key storage resource",
8646     "type": "integer"
8647 },
8648 },
8649 "required": [
8650     "encoding"
8651 ],
8652 "type": "object"
8653 },
8654 "publicdata": {
8655     "description": "Public credential information",
8656     "properties": {
8657         "data": {
8658             "description": "The encoded value",
8659             "maxLength": 3072,
8660             "type": "string"
8661         },
8662         "encoding": {
8663             "description": "A string specifying the encoding format of the data contained in
8664 the pubdata",
8665             "x-detail-desc": [
8666                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8667                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8668                 "oic.sec.encoding.base64 - Base64 encoded object",
8669                 "oic.sec.encoding.uri - URI reference",
8670                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8671                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
8672                 "oic.sec.encoding.raw - Raw hex encoded data"
8673             ],
8674             "enum": [
8675                 "oic.sec.encoding.jwt",
8676                 "oic.sec.encoding.cwt",
8677                 "oic.sec.encoding.base64",
8678                 "oic.sec.encoding.uri",
8679                 "oic.sec.encoding.pem",
8680                 "oic.sec.encoding.der",
8681                 "oic.sec.encoding.raw"
8682             ],
8683             "type": "string"
8684         }
8685     },
8686     "type": "object"
8687 },
8688 "roleid": {
8689     "description": "The role this credential possesses\nSecurity role specified as an
8690 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
8691     "properties": {
8692         "authority": {
8693             "description": "The Authority component of the entity being identified. A NULL
8694 <Authority> refers to the local entity or device.",
8695             "type": "string"
8696         },
8697         "role": {

```

```

8698         "description": "The ID of the role being identified.",
8699         "type": "string"
8700     },
8701 },
8702     "required": [
8703         "role"
8704     ],
8705     "type": "object"
8706 },
8707     "subjectuuid": {
8708         "anyOf": [
8709             {
8710                 "description": "The id of the device, which the cred entry applies to or \"*\"
8711 for wildcard identity",
8712                 "pattern": "^\\*$",
8713                 "type": "string"
8714             },
8715             {
8716                 "description": "Format pattern according to IETF RFC 4122.",
8717                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8718 F0-9]{12}$",
8719                 "type": "string"
8720             }
8721         ]
8722     },
8723     "type": "object"
8724 },
8725     "type": "array"
8726 },
8727 },
8728 },
8729     "type": "object",
8730     "required": ["roles"]
8731 }
8732 }
8733 }
8734

```

### 8735 C.11.5 Property definition

8736 Table C.20 defines the Properties that are part of the ['oic.r.roles'] Resource Type

8737 **Table C.18 – The Property definitions of the Resource with type 'rt' = ['oic.r.roles']**

Property name	Value type	Mandatory	Access mode	Description
roles	array: see schema	Yes	Read Write	List of role certificates
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
roles	array: see schema	Yes	Read Write	List of role certificates
id	multiple types: see schema	No	Read Write	

### 8738 C.11.6 CRUDN behaviour

8739 Table C.21 defines the CRUDN operations that are supported on the ['oic.r.roles'] Resource Type

8740 **Table C.19 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.roles']**

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

	get	post	delete	observe
--	-----	------	--------	---------

8741 **C.12 Signed Access Control List**

8742 **C.12.1 Introduction**

8743 This Resource specifies a signed ACL object.

8744

8745 **C.12.2 Well-known URI**

8746 /oic/sec/sacl

8747 **C.12.3 Resource type**

8748 The resource type (rt) is defined as: ['oic.r.sacl'].

8749 **C.12.4 OpenAPI 2.0 definition**

```

8750 {
8751   "swagger": "2.0",
8752   "info": {
8753     "title": "Signed Access Control List",
8754     "version": "v1.0-20150819",
8755     "license": {
8756       "name": "OCF Data Model License",
8757       "url":
8758         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8759 CENSE.md",
8760       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8761 reserved."
8762     },
8763     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8764   },
8765   "schemes": ["http"],
8766   "consumes": ["application/json"],
8767   "produces": ["application/json"],
8768   "paths": {
8769     "/oic/sec/sacl" : {
8770       "get": {
8771         "description": "This Resource specifies a signed ACL object.\n",
8772         "parameters": [
8773           { "$ref": "#/parameters/interface" }
8774         ],
8775         "responses": {
8776           "200": {
8777             "description": "",
8778             "x-example":
8779               {
8780                 "rt": ["oic.r.sacl"],
8781                 "aclist2": [
8782                   {
8783                     "aceid": 1,
8784                     "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8785                     "resources": [
8786                       {
8787                         "href": "/temp",
8788                         "rt": ["oic.r.temperature"],
8789                         "if": ["oic.if.baseline", "oic.if.a"]
8790                       },
8791                       {
8792                         "href": "/temp",
8793                         "rt": ["oic.r.temperature"],
8794                         "if": ["oic.if.baseline", "oic.if.s"]
8795                       }
8796                     ]
8797                   },
8798                   "permission": 31,
8799                   "validity": [
8800                     {
8801                       "period": "20160101T180000Z/20170102T070000Z",

```

```

8802 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8803     },
8804     {
8805         "period": "20160101T180000Z/PT5H30M",
8806         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8807     }
8808     ],
8809 },
8810 {
8811     "aceid": 2,
8812     "subject": {
8813         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8814         "role": "SOME_STRING"
8815     },
8816     "resources": [
8817         {
8818             "href": "/light",
8819             "rt": ["oic.r.light"],
8820             "if": ["oic.if.baseline", "oic.if.a"]
8821         },
8822         {
8823             "href": "/door",
8824             "rt": ["oic.r.door"],
8825             "if": ["oic.if.baseline", "oic.if.a"]
8826         }
8827     ],
8828     "permission": 15
8829 },
8830 ],
8831 "signature": {
8832     "sigtype": "oic.sec.sigtype.pk7",
8833     "sigvalue": "ENCODED-SIGNATURE-VALUE"
8834 },
8835 },
8836 "schema": { "$ref": "#/definitions/Sacl" }
8837 }
8838 },
8839 },
8840 "post": {
8841     "description": "Sets the sacl resource data\n",
8842     "parameters": [
8843         { "$ref": "#/parameters/interface",
8844         {
8845             "name": "body",
8846             "in": "body",
8847             "required": true,
8848             "schema": { "$ref": "#/definitions/Sacl" },
8849             "x-example":
8850             {
8851                 "aclist2": [
8852                     {
8853                         "aceid": 1,
8854                         "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8855                         "resources": [
8856                             {
8857                                 "href": "/temp",
8858                                 "rt": ["oic.r.temperature"],
8859                                 "if": ["oic.if.baseline", "oic.if.a"]
8860                             },
8861                             {
8862                                 "href": "/temp",
8863                                 "rt": ["oic.r.temperature"],
8864                                 "if": ["oic.if.baseline", "oic.if.s"]
8865                             }
8866                         ],
8867                         "permission": 31,
8868                         "validity": [
8869                             {
8870                                 "period": "20160101T180000Z/20170102T070000Z",
8871                                 "recurrence": [ "DSTART:XXXXX",
8872 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]

```

```

8873     },
8874     {
8875         "period": "20160101T180000Z/PT5H30M",
8876         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8877     }
8878 ]
8879 },
8880 {
8881     "aceid": 2,
8882     "subject": {
8883         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8884         "role": "SOME_STRING"
8885     },
8886     "resources": [
8887         {
8888             "href": "/light",
8889             "rt": ["oic.r.light"],
8890             "if": ["oic.if.baseline", "oic.if.a"]
8891         },
8892         {
8893             "href": "/door",
8894             "rt": ["oic.r.door"],
8895             "if": ["oic.if.baseline", "oic.if.a"]
8896         }
8897     ],
8898     "permission": 15
8899 }
8900 ],
8901 "signature": {
8902     "sigtype": "oic.sec.sigtype.pk7",
8903     "sigvalue": "ENCODED-SIGNATURE-VALUE"
8904 }
8905 }
8906 ],
8907 },
8908 "responses": {
8909     "400": {
8910         "description": "The request is invalid."
8911     },
8912     "201": {
8913         "description": "The ACL entry is created."
8914     },
8915     "204": {
8916         "description": "The ACL entry is updated."
8917     }
8918 },
8919 },
8920 "put": {
8921     "description": "Sets the sacl resource data\n",
8922     "parameters": [
8923         { "$ref": "#/parameters/interface" },
8924         {
8925             "name": "body",
8926             "in": "body",
8927             "required": true,
8928             "schema": { "$ref": "#/definitions/Sacl" },
8929             "x-example":
8930             {
8931                 "aclist2": [
8932                     {
8933                         "aceid": 1,
8934                         "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8935                         "resources": [
8936                             {
8937                                 "href": "/temp",
8938                                 "rt": ["oic.r.temperature"],
8939                                 "if": ["oic.if.baseline", "oic.if.a"]
8940                             },
8941                             {
8942                                 "href": "/temp",
8943                                 "rt": ["oic.r.temperature"],

```

```

8944         "if": ["oic.if.baseline", "oic.if.s"]
8945     }
8946 ],
8947 "permission": 31,
8948 "validity": [
8949     {
8950         "period": "20160101T180000Z/20170102T070000Z",
8951         "recurrence": [ "DSTART:XXXXX",
8952 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8953     },
8954     {
8955         "period": "20160101T180000Z/PT5H30M",
8956         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8957     }
8958 ],
8959 },
8960 {
8961     "aceid": 2,
8962     "subject": {
8963         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8964         "role": "SOME_STRING"
8965     },
8966     "resources": [
8967         {
8968             "href": "/light",
8969             "rt": ["oic.r.light"],
8970             "if": ["oic.if.baseline", "oic.if.a"]
8971         },
8972         {
8973             "href": "/door",
8974             "rt": ["oic.r.door"],
8975             "if": ["oic.if.baseline", "oic.if.a"]
8976         }
8977     ],
8978     "permission": 15
8979 }
8980 ],
8981 "signature": {
8982     "sigtype": "oic.sec.sigtype.pk7",
8983     "sigvalue": "ENCODED-SIGNATURE-VALUE"
8984 }
8985 }
8986 },
8987 ],
8988 "responses": {
8989     "400": {
8990         "description": "The request is invalid."
8991     },
8992     "201": {
8993         "description": "The signed ACL entry is created."
8994     }
8995 }
8996 },
8997 "delete": {
8998     "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
8999 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
9000 specified, only the matched entry is deleted.\n",
9001     "parameters": [
9002         { "$ref": "#/parameters/interface",
9003           {
9004             "in": "query",
9005             "description": "Delete the signed ACL identified by the string containing subject
9006 UUID.\n",
9007             "type": "string",
9008             "name": "subject"
9009           }
9010         },
9011         {
9012             "description": "The signed ACL instance or the the entire signed ACL resource has
9013             been successfully deleted."
9014         }

```

```

9015     },
9016     "400": {
9017         "description": "The request is invalid."
9018     }
9019 }
9020 }
9021 },
9022 },
9023 "parameters": {
9024     "interface": {
9025         "in": "query",
9026         "name": "if",
9027         "type": "string",
9028         "enum": ["oic.if.baseline"]
9029     }
9030 },
9031 "definitions": {
9032     "Sacl": {
9033         "properties": {
9034             "rt": {
9035                 "description": "Resource Type of the Resource",
9036                 "items": {
9037                     "maxLength": 64,
9038                     "type": "string",
9039                     "enum": ["oic.r.sacl"]
9040                 },
9041                 "minItems": 1,
9042                 "readOnly": true,
9043                 "type": "array"
9044             },
9045             "aclist2": {
9046                 "description": "Access Control Entries in the Acl resource",
9047                 "items": {
9048                     "properties": {
9049                         "aceid": {
9050                             "description": "An identifier for the ACE that is unique within the ACL. In cases
9051 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
9052                             "minimum": 1,
9053                             "type": "integer"
9054                         },
9055                         "permission": {
9056                             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
9057 permissions",
9058                             "x-detail-desc": [
9059                                 "0 - No permissions",
9060                                 "1 - Create permission is granted",
9061                                 "2 - Read, observe, discover permission is granted",
9062                                 "4 - Write, update permission is granted",
9063                                 "8 - Delete permission is granted",
9064                                 "16 - Notify permission is granted"
9065                             ],
9066                             "maximum": 31,
9067                             "minimum": 0,
9068                             "type": "integer"
9069                         },
9070                         "resources": {
9071                             "description": "References the application's resources to which a security policy
9072 applies",
9073                             "items": {
9074                                 "description": "Each resource must have at least one of these properties set",
9075                                 "properties": {
9076                                     "href": {
9077                                         "allOf": [
9078                                             {
9079                                                 "description": "When present, the ACE only applies when the href matches"
9080                                             },
9081                                             {
9082                                                 "description": "This is the target URI, it can be specified as a Relative
9083 Reference or fully-qualified URI.",
9084                                                 "format": "uri",
9085                                                 "maxLength": 256,

```

```

9086         "type": "string"
9087     }
9088 ]
9089 },
9090 "if": {
9091     "description": "When present, the ACE only applies when the if (interface)
9092 matches\nThe interface set supported by this resource",
9093     "items": {
9094         "enum": [
9095             "oic.if.baseline",
9096             "oic.if.ll",
9097             "oic.if.b",
9098             "oic.if.rw",
9099             "oic.if.r",
9100             "oic.if.a",
9101             "oic.if.s"
9102         ],
9103         "type": "string"
9104     },
9105     "minItems": 1,
9106     "type": "array"
9107 },
9108 "rt": {
9109     "description": "When present, the ACE only applies when the rt (resource type)
9110 matches\nResource Type of the Resource",
9111     "items": {
9112         "maxLength": 64,
9113         "type": "string"
9114     },
9115     "minItems": 1,
9116     "type": "array"
9117 },
9118 "wc": {
9119     "description": "A wildcard matching policy",
9120     "pattern": "^[~+]*$",
9121     "type": "string"
9122 }
9123 },
9124 "type": "object"
9125 },
9126 "type": "array"
9127 },
9128 "subject": {
9129     "anyOf": [
9130         {
9131             "description": "Device identifier",
9132             "properties": {
9133                 "uuid": {
9134                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
9135 4122.",
9136                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
9137 fA-F0-9]{12}$",
9138                     "type": "string"
9139                 }
9140             },
9141             "required": [
9142                 "uuid"
9143             ],
9144             "type": "object"
9145         },
9146         {
9147             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
9148 <Authority> refers to the local entity or device.",
9149             "properties": {
9150                 "authority": {
9151                     "description": "The Authority component of the entity being identified. A
9152 NULL <Authority> refers to the local entity or device.",
9153                     "type": "string"
9154                 },
9155                 "role": {
9156                     "description": "The ID of the role being identified.",

```

```

9157         "type": "string"
9158     },
9159     "required": [
9160         "role"
9161     ],
9162     "type": "object"
9163 },
9164 {
9165     "properties": {
9166         "conntype": {
9167             "description": "This property allows an ACE to be matched based on the
9168 connection or message type",
9169             "x-detail-desc": [
9170                 "auth-crypt - ACE applies if the Client is authenticated and the data
9171 channel or message is encrypted and integrity protected",
9172                 "anon-clear - ACE applies if the Client is not authenticated and the data
9173 channel or message is not encrypted but may be integrity protected"
9174             ],
9175             "enum": [
9176                 "auth-crypt",
9177                 "anon-clear"
9178             ],
9179             "type": "string"
9180         }
9181     },
9182     "required": [
9183         "conntype"
9184     ],
9185     "type": "object"
9186 }
9187 ],
9188 },
9189 "validity": {
9190     "description": "validity is an array of time-pattern objects",
9191     "items": {
9192         "description": "The time-pattern contains a period and recurrence expressed in
9193 IETF RFC 5545 syntax",
9194         "properties": {
9195             "period": {
9196                 "description": "String represents a period using the IETF RFC 5545 Period",
9197                 "type": "string"
9198             },
9199             "recurrence": {
9200                 "description": "String array represents a recurrence rule using the IETF RFC
9201 5545 Recurrence",
9202                 "items": {
9203                     "type": "string"
9204                 },
9205                 "type": "array"
9206             }
9207         },
9208         "required": [
9209             "period"
9210         ],
9211         "type": "object"
9212     },
9213     "type": "array"
9214 },
9215 },
9216 "required": [
9217     "aceid",
9218     "resources",
9219     "permission",
9220     "subject"
9221 ],
9222 "type": "object"
9223 },
9224 "type": "array"
9225 },
9226 "n": {
9227

```

```

9228     "$ref":
9229     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9230     schema.json#/definitions/n"
9231     },
9232     "id": {
9233     "$ref":
9234     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9235     schema.json#/definitions/id"
9236     },
9237     "signature": {
9238     "description": "The signature over the ACL resource\nEncoded signature data",
9239     "properties": {
9240     "sigtype": {
9241     "description": "The string specifies the predefined signature format",
9242     "x-detail-desc": [
9243     "IETF RFC 7515 JSON web signature (JWS) object",
9244     "IETF RFC 2315 base64 encoded object",
9245     "CBOR encoded JWS object"
9246     ],
9247     "enum": [
9248     "oic.sec.sigtype.jws",
9249     "oic.sec.sigtype.pk7",
9250     "oic.sec.sigtype.cws"
9251     ],
9252     "type": "string"
9253     },
9254     "sigvalue": {
9255     "description": "The encoded signature",
9256     "type": "string"
9257     }
9258     },
9259     "required": [
9260     "sigtype",
9261     "sigvalue"
9262     ],
9263     "type": "object"
9264     },
9265     "if": {
9266     "description": "The interface set supported by this resource",
9267     "items": {
9268     "enum": [
9269     "oic.if.baseline"
9270     ],
9271     "type": "string"
9272     },
9273     "minItems": 1,
9274     "readOnly": true,
9275     "type": "array"
9276     }
9277     },
9278     "type": "object",
9279     "required": ["aclist2", "signature"]
9280     }
9281     }
9282     }
9283

```

### 9284 C.12.5 Property definition

9285 Table C.22 defines the Properties that are part of the ['oic.r.sacl'] Resource Type

9286 **Table C.20 – The Property definitions of the Resource with type 'rt' = ['oic.r.sacl']**

Property name	Value type	Mandatory	Access mode	Description
id	multiple types: see schema	No	Read Write	
rt	array: schema	No	Read Only	Resource Type of the Resource

aclist2	array: schema	see	Yes	Read Write	Access Control Entries in the Acl resource
signature	object: schema	see	Yes	Read Write	The signature over the ACL resource Encoded signature data
n	multiple types: see schema		No	Read Write	
if	array: schema	see	No	Read Only	The OCF Interface set supported by this resource

9287 **C.12.6 CRUDN behaviour**

9288 Table C.23 defines the CRUDN operations that are supported on the ['oic.r.sacl'] Resource Type

9289 **Table C.21 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sacl']**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

9290 **C.13 Session**

9291 **C.13.1 Introduction**

9292 Resource that manages the persistent session between a Device and OCF Cloud

9293 **C.13.2 Well-known URI**

9294 /oic/sec/session

9295 **C.13.3 Resource type**

9296 The resource type (rt) is defined as: ['oic.r.session'].

9297 **C.13.4 OpenAPI 2.0 definition**

```

9298 {
9299   "swagger": "2.0",
9300   "info": {
9301     "title": "Session",
9302     "version": "v1.0-20181001",
9303     "license": {
9304       "name": "OCF Data Model License",
9305       "url":
9306         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9307         CENSE.md",
9308       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9309         reserved."
9310     },
9311     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9312   },
9313   "schemes": ["http"],
9314   "consumes": ["application/json"],
9315   "produces": ["application/json"],
9316   "paths": {
9317     "/oic/sec/session" : {
9318       "post": {
9319         "description": "Resource that manages the persistent session between a Device and OCF
9320         Cloud",
9321         "parameters": [
9322           { "$ref": "#/parameters/interface" },
9323           {
9324             "name": "body",

```

```

9325         "in": "body",
9326         "required": true,
9327         "schema": { "$ref": "#/definitions/Account-Session-Request" },
9328         "x-example":
9329             {
9330                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9331                 "di" : "9cfbeb8e-5a1e-4d1c-9d01-00c04fd430c8",
9332                 "accesstoken" : "0f3d9f7fe5491d54077d",
9333                 "login" : true
9334             }
9335     },
9336 ],
9337 "responses": {
9338     "204": {
9339         "description" : "",
9340         "x-example":
9341             {
9342                 "rt": ["oic.r.session"],
9343                 "expiresin" : 3600
9344             },
9345         "schema": { "$ref": "#/definitions/Account-Session-Response" }
9346     }
9347 },
9348 },
9349 },
9350 },
9351 "parameters": {
9352     "interface": {
9353         "in" : "query",
9354         "name" : "if",
9355         "type" : "string",
9356         "enum" : ["oic.if.baseline"]
9357     }
9358 },
9359 "definitions": {
9360     "Account-Session-Request" : {
9361         "properties": {
9362             "uid": {
9363                 "description": "Format pattern according to IETF RFC 4122.",
9364                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9365 9]{12}$",
9366                 "type": "string"
9367             },
9368             "di": {
9369                 "description": "The device ID\nFormat pattern according to IETF RFC 4122.",
9370                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9371 9]{12}$",
9372                 "type": "string"
9373             },
9374             "accesstoken": {
9375                 "description": "Access-Token used to grant access right for the device to sign-in",
9376                 "pattern": "(?!$|\\s+).*",
9377                 "type": "string"
9378             },
9379             "login": {
9380                 "description": "Action for the request: true = login, false = logout",
9381                 "type": "boolean"
9382             }
9383         },
9384         "type" : "object",
9385         "required": ["uid", "di", "accesstoken", "login"]
9386     },
9387     "Account-Session-Response" : {
9388         "properties": {
9389             "expiresin": {
9390                 "description": "Access-Token remaining life time in seconds (-1 if permanent)",
9391                 "readOnly": true,
9392                 "type": "integer"
9393             },
9394             "rt": {
9395                 "description": "Resource Type of the Resource",

```

```

9396     "items": {
9397         "maxLength": 64,
9398         "type": "string",
9399         "enum": ["oic.r.session"]
9400     },
9401     "minItems": 1,
9402     "readOnly": true,
9403     "type": "array"
9404 },
9405 "n": {
9406     "$ref":
9407 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9408 schema.json#/definitions/n"
9409 },
9410 "id": {
9411     "$ref":
9412 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9413 schema.json#/definitions/id"
9414 },
9415 "if": {
9416     "description": "The interface set supported by this resource",
9417     "items": {
9418         "enum": [
9419             "oic.if.baseline"
9420         ],
9421         "type": "string"
9422     },
9423     "minItems": 1,
9424     "readOnly": true,
9425     "type": "array"
9426 }
9427 },
9428 "type" : "object",
9429 "required" : ["expiresin"]
9430 }
9431 }
9432 }
9433 }

```

### 9434 C.13.5 Property definition

9435 Table C.24 defines the Properties that are part of the ['oic.r.session'] Resource Type

9436 **Table C.22 – The Property definitions of the Resource with type 'rt' = ['oic.r.session']**

Property name	Value type	Mandatory	Access mode	Description
di	string	Yes	Read Write	The device ID Format pattern according to IETF RFC 4122.
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
login	boolean	Yes	Read Write	Action for the request: true = login, false = logout
accesstoken	string	Yes	Read Write	Access-Token used to grant access right for the device to sign-in
expiresin	integer	Yes	Read Only	Access-Token remaining life

				time in seconds (-1 if permanent)
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource

9437 **C.13.6 CRUDN behaviour**

9438 Table C.25 defines the CRUDN operations that are supported on the ['oic.r.session'] Resource  
9439 Type

9440 **Table C.23 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.session']**

Create	Read	Update	Delete	Notify
		post		

9441 **C.14 Security Profile**

9442 **C.14.1 Introduction**

9443 Resource specifying supported and active security profile(s)

9444

9445 **C.14.2 Well-known URI**

9446 /oic/sec/sp

9447 **C.14.3 Resource type**

9448 The resource type (rt) is defined as: ['oic.r.sp'].

9449 **C.14.4 OpenAPI 2.0 definition**

```
9450 {
9451   "swagger": "2.0",
9452   "info": {
9453     "title": "Security Profile",
9454     "version": "v1.0-20190208",
9455     "license": {
9456       "name": "OCF Data Model License",
9457       "url":
9458 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9459 CENSE.md",
9460       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9461 reserved."
9462     },
9463     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9464   },
9465   "schemes": ["http"],
9466   "consumes": ["application/json"],
9467   "produces": ["application/json"],
9468   "paths": {
9469     "/oic/sec/sp": {
9470       "get": {
9471         "description": "Resource specifying supported and active security profile(s)\n",
9472         "parameters": [
9473           {"$ref": "#/parameters/interface"}
9474         ],
9475         "responses": {
9476           "200": {
```

```

9477         "description" : "",
9478         "x-example":
9479         {
9480           "rt": ["oic.r.sp"],
9481           "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9482           "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9483         },
9484         "schema": { "$ref": "#/definitions/SP" }
9485       },
9486       "400": {
9487         "description" : "The request is invalid."
9488       }
9489     },
9490   },
9491   "post": {
9492     "description": "Sets or updates device provisioning status data.\n",
9493     "parameters": [
9494       { "$ref": "#/parameters/interface",
9495         {
9496           "name": "body",
9497           "in": "body",
9498           "required": true,
9499           "schema": { "$ref": "#/definitions/SP-Update" } },
9500       "x-example":
9501       {
9502         "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9503         "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9504       }
9505     ],
9506   },
9507   "responses": {
9508     "200": {
9509       "description" : "",
9510       "x-example":
9511       {
9512         "rt": ["oic.r.sp"],
9513         "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9514         "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9515       },
9516       "schema": { "$ref": "#/definitions/SP" }
9517     },
9518     "400": {
9519       "description" : "The request is invalid."
9520     }
9521   }
9522 },
9523 },
9524 },
9525 "parameters": {
9526   "interface" : {
9527     "in" : "query",
9528     "name" : "if",
9529     "type" : "string",
9530     "enum" : ["oic.if.baseline"]
9531   }
9532 },
9533 "definitions": {
9534   "SP" : {
9535     "properties": {
9536       "rt": {
9537         "description": "Resource Type of the Resource",
9538         "items": {
9539           "maxLength": 64,
9540           "type": "string",
9541           "enum": ["oic.r.sp"]
9542         },
9543         "minItems": 1,
9544         "readOnly": true,
9545         "type": "array"
9546       },
9547       "n": {

```

```

9548     "$ref":
9549     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9550     schema.json#/definitions/n"
9551     },
9552     "id": {
9553     "$ref":
9554     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9555     schema.json#/definitions/id"
9556     },
9557     "currentprofile": {
9558     "description": "Security Profile currently active",
9559     "type": "string"
9560     },
9561     "supportedprofiles": {
9562     "description": "Array of supported Security Profiles",
9563     "items": {
9564     "type": "string"
9565     },
9566     "type": "array"
9567     },
9568     "if": {
9569     "description": "The interface set supported by this resource",
9570     "items": {
9571     "enum": [
9572     "oic.if.baseline"
9573     ],
9574     "type": "string"
9575     },
9576     "minItems": 1,
9577     "readOnly": true,
9578     "type": "array"
9579     }
9580     },
9581     "type": "object",
9582     "required": ["supportedprofiles", "currentprofile"]
9583     },
9584     "SP-Update" : {
9585     "properties": {
9586     "currentprofile": {
9587     "description": "Security Profile currently active",
9588     "type": "string"
9589     },
9590     "supportedprofiles": {
9591     "description": "Array of supported Security Profiles",
9592     "items": {
9593     "type": "string"
9594     },
9595     "type": "array"
9596     }
9597     },
9598     "type": "object"
9599     }
9600     }
9601     }
9602
9603

```

#### 9604 C.14.5 Property definition

9605 Table C.26 defines the Properties that are part of the ['oic.r.sp'] Resource Type

9606 **Table C.24 – The Property definitions of the Resource with type 'rt' = ['oic.r.sp']**

Property name	Value type	Mandatory	Access mode	Description
supportedprofiles	array: see schema		Read Write	Array of supported Security Profiles
currentprofile	string		Read Write	Security Profile currently active

id	multiple types: see schema	No	Read Write	
rt	array: schema see	No	Read Only	Resource Type of the Resource
if	array: schema see	No	Read Only	The interface set supported by this resource
n	multiple types: see schema	No	Read Write	

9607

#### 9608 C.14.6 CRUDN behaviour

9609 Table C.27 defines the CRUDN operations that are supported on the ['oic.r.sp'] Resource Type

9610

**Table C.25 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sp']**

Create	Read	Update	Delete	Notify
	get	post		observe

#### 9611 C.15 Token Refresh

##### 9612 C.15.1 Introduction

9613 Obtain fresh access-token using the refresh token, client should refresh access-token before it  
9614 expires.

9615

##### 9616 C.15.2 Well-known URI

9617 /oic/sec/tokenrefresh

##### 9618 C.15.3 Resource type

9619 The resource type (rt) is defined as: ['oic.r.tokenrefresh'].

##### 9620 C.15.4 OpenAPI 2.0 definition

```

9621 {
9622   "swagger": "2.0",
9623   "info": {
9624     "title": "Token Refresh",
9625     "version": "v1.0-20181001",
9626     "license": {
9627       "name": "OCF Data Model License",
9628       "url":
9629 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9630 CENSE.md",
9631       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9632 reserved."
9633     },
9634     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9635   },
9636   "schemes": ["http"],
9637   "consumes": ["application/json"],
9638   "produces": ["application/json"],
9639   "paths": {
9640     "/oic/sec/tokenrefresh" : {
9641       "post": {
9642         "description": "Obtain fresh access-token using the refresh token, client should refresh
9643 access-token before it expires.\n",
9644         "parameters": [
9645           {"$ref": "#/parameters/interface"},
9646           {
9647             "name": "body",
9648             "in": "body",
9649             "required": true,

```

```

9650         "schema": { "$ref": "#/definitions/TokenRefresh-Request" },
9651         "x-example":
9652         {
9653             "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9654             "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
9655             "refreshtoken" : "00fe4644a6fbe5324eec"
9656         }
9657     },
9658     "responses": {
9659         "204": {
9660             "description": "2.04 Changed respond with new access-token\n",
9661             "x-example":
9662             {
9663                 "rt": ["oic.r.tokenrefresh"],
9664                 "accesstoken" : "8ce598980761869837be",
9665                 "refreshtoken" : "d4922312b6df0518e146",
9666                 "expiresin" : 3600
9667             },
9668             "schema": { "$ref": "#/definitions/TokenRefresh-Response" }
9669         }
9670     }
9671 }
9672 }
9673 }
9674 },
9675 },
9676 "parameters": {
9677     "interface": {
9678         "in": "query",
9679         "name": "if",
9680         "type": "string",
9681         "enum": ["oic.if.baseline"]
9682     }
9683 },
9684 "definitions": {
9685     "TokenRefresh-Request" : {
9686         "properties": {
9687             "refreshtoken": {
9688                 "description": "Refresh token received by account management or during token refresh
9689 procedure",
9690                 "pattern": "(?!$|\\s+).*",
9691                 "type": "string"
9692             },
9693             "uid": {
9694                 "description": "Format pattern according to IETF RFC 4122.",
9695                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9696 9]{12}$",
9697                 "type": "string"
9698             },
9699             "di": {
9700                 "description": "Format pattern according to IETF RFC 4122.",
9701                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9702 9]{12}$",
9703                 "type": "string"
9704             }
9705         },
9706         "type": "object",
9707         "required": ["uid", "di", "refreshtoken"]
9708     },
9709     "TokenRefresh-Response" : {
9710         "properties": {
9711             "expiresin": {
9712                 "description": "Access-Token life time in seconds (-1 if permanent)",
9713                 "readOnly": true,
9714                 "type": "integer"
9715             },
9716             "rt": {
9717                 "description": "Resource Type of the Resource",
9718                 "items": {
9719                     "maxLength": 64,
9720                     "type": "string",

```

```

9721         "enum": ["oic.r.tokenrefresh"]
9722     },
9723     "minItems": 1,
9724     "readOnly": true,
9725     "type": "array"
9726 },
9727 "refreshToken": {
9728     "description": "Refresh token received by account management or during token refresh
9729 procedure",
9730     "pattern": "(?!$|\\s+).*",
9731     "type": "string"
9732 },
9733 "accessToken": {
9734     "description": "Granted Access-Token",
9735     "pattern": "(?!$|\\s+).*",
9736     "readOnly": true,
9737     "type": "string"
9738 },
9739 "n": {
9740     "$ref":
9741 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9742 schema.json#/definitions/n"
9743 },
9744 "id": {
9745     "$ref":
9746 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9747 schema.json#/definitions/id"
9748 },
9749 "if" :
9750 {
9751     "description": "The interface set supported by this resource",
9752     "items": {
9753         "enum": [
9754             "oic.if.baseline"
9755         ],
9756         "type": "string"
9757     },
9758     "minItems": 1,
9759     "readOnly": true,
9760     "type": "array"
9761 }
9762 },
9763 "type" : "object",
9764 "required": ["accessToken", "refreshToken", "expiresin"]
9765 }
9766 }
9767 }
9768

```

9769 **C.15.5 Property definition**

9770 Table C.28 defines the Properties that are part of the ['oic.r.tokenrefresh'] Resource Type

9771 **Table C.26 – The Property definitions of the Resource with type 'rt' = ['oic.r.tokenrefresh']**

Property name	Value type	Mandatory	Access mode	Description
accessToken	string	Yes	Read Only	Granted Access-Token
n	multiple types: see schema	No	Read Write	
refreshToken	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure

id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
expiresin	integer	Yes	Read Only	Access-Token life time in seconds (-1 if permanent)
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
refreshtoken	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.

9772 **C.15.6 CRUDN behaviour**

9773 Table C.29 defines the CRUDN operations that are supported on the ['oic.r.tokenrefresh'] Resource  
9774 Type

9775 **Table C.27 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.tokenrefresh']**

Create	Read	Update	Delete	Notify
		post		

9776

9777

9778

9779  
9780  
9781  
9782

**Annex D  
(informative)**

**OID definitions**

9783 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to  
9784 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of  
9785 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
9786 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
9787     private(4) enterprise(1) OCF(51414) }
9788
9789 -- OCF Security specific OIDs
9790
9791 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
9792 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9793
9794 -- OCF Security Categories
9795
9796 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
9797 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
9798
9799 -- OCF Security Profiles
9800
9801 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
9802 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
9803 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
9804 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
9805 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
9806
9807 sp-unspecified-v0 ::= ocfSecurityProfileOID {id-sp-unspecified 0}
9808 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
9809 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
9810 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
9811 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
9812
9813 ocfSecurityProfileOID ::= UTF8String
9814
9815 -- OCF Security Certificate Policies
9816
9817 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
9818
9819 -- OCF X.509v3 Extensions
9820
9821 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9822 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
9823 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
9824 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
9825
9826 ocfVersion ::= SEQUENCE {
9827     major    INTEGER,
9828     minor    INTEGER,
9829     build    INTEGER}
9830
9831 ocfCompliance ::= SEQUENCE {
9832     version        ocfVersion,
9833     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
9834     deviceName     UTF8String,
9835     deviceManufacturer UTF8String}
9836
9837 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
9838 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
9839
9840 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
9841
9842 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
9843
9844 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
9845 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
9846 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
9847
9848 ocfCPLAttributes ::= SEQUENCE {
9849     cpl-at-IANAPen UTF8String,
9850     cpl-at-model UTF8String,
9851     cpl-at-version UTF8String}
```

9852 **Annex E**  
 9853 **(informative)**

9854 **Security considerations specific to Bridged Protocols**  
 9855

9856 The text in this Annex is provided for information only. This Annex has no normative impact. This  
 9857 information is applicable at the time of initial publication and may become out of date.

9858 **E.1 Security Considerations specific to the AllJoyn Protocol**

9859 This clause intentionally left empty.

9860 **E.2 Security Considerations specific to the Bluetooth LE Protocol**

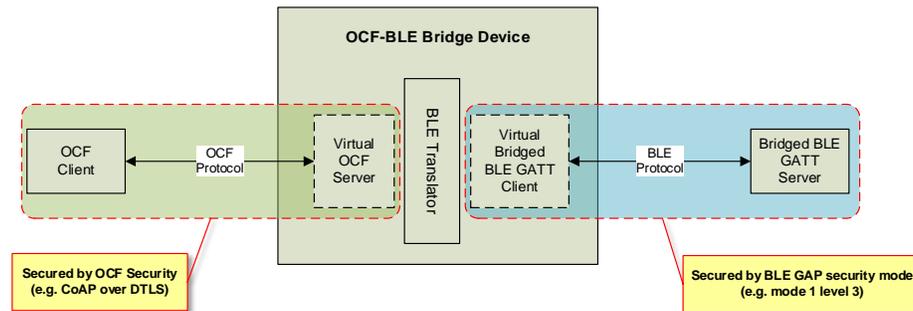
9861 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security mode  
 9862 has several security levels (see Table E.1)

9863 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF  
 9864 perspective. The appropriate selection of security mode and level is left to the vendor.

9865 **Table E.1 GAP security mode**

GAP security mode	security level
Security mode 1	1 (no security)
	2 (Unauthenticated pairing with encryption)
	3 (Authenticated pairing with encryption)
	4 (Authenticated LE Secure Connections pairing with encryption)
Security mode 2	1 (Unauthenticated pairing with data signing)
	2 (Authenticated pairing with data signing)

9866 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge device are secured by  
 9867 their own security.



9868 **Figure E-1 Security Considerations for BLE Bridge**  
 9869

9870 **E.3 Security Considerations specific to the oneM2M Protocol**

9871 This clause intentionally left empty.

9872 **E.4 Security Considerations specific to the U+ Protocol**

9873 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

**Table E.2 TLS 1.2 Cipher Suites used by U+**

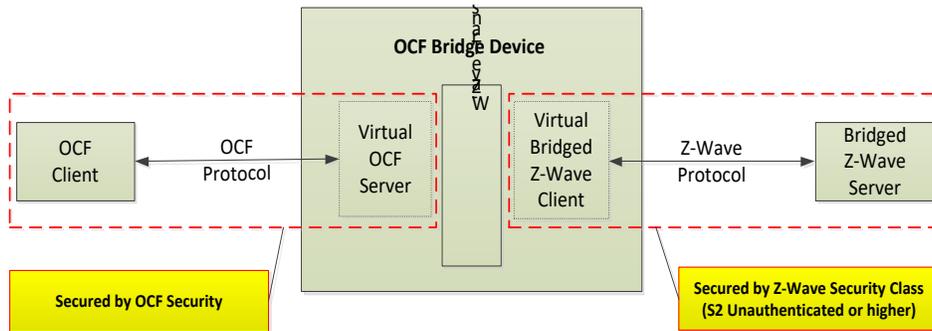
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_CCM
TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CCM
TLS_DHE_RSA_WITH_AES_256_CCM_8

9875 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

#### 9876 **E.5 Security Considerations specific to the Z-Wave Protocol**

9877 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2 Security  
 9878 Class, as shown in Table 7 below.. Bridged Z-wave Servers using S2 Security Class for  
 9879 communication with a Virtual Bridged Client would typically be considered secure from an OCF  
 9880 perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

9881 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their  
 9882 own security.



9883

9884

**Figure E-2 Security Considerations for Z-Wave Bridge**

9885 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2  
 9886 Unauthenticated provides the following advantages from the security perspective;

- 9887 – The unique device specific key for every secure device enables validation of device identity and  
 9888 prevents man-in-the-middle compromises to security
- 9889 – The Secure cryptographic key exchange methods during inclusion achieves high level of  
 9890 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 9891 – Out of band key exchange for product authentication which is combined with device specific  
 9892 key prevents eavesdropping and man-in-the-middle attack vectors.

9893 See Table E.3 for a summary of Z-Wave Security Classes.

9894

**Table E.3 Z-Wave Security Class**

Security Class	Class Name	Validation of device identity	Key Exchange	Message Encapsulation
S2	S2 Access Control	Device Specific key	Out-of-band inclusion	Encrypted command transmission
	S2 Authenticated	Device Specific key	Out-of-band inclusion	Encrypted command transmission
	S2 Unauthenticated	Device Specific key	Z-wave RF band used for inclusion	Encrypted command transmission
S0	S0 Authenticated	N/A	Z-wave RF band used for inclusion	Encrypted command transmission

9895 On the other hand, S0 Security Class has the vulnerability of security during inclusion by  
 9896 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the  
 9897 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices  
 9898 might be no longer secure in that case.

9899 **E.6 Security Considerations specific to the Zigbee Protocol**

9900 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the  
 9901 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0  
 9902 stack, nwkSecurityLevel, represents the security level of a device. Further details can be found in  
 9903 the .

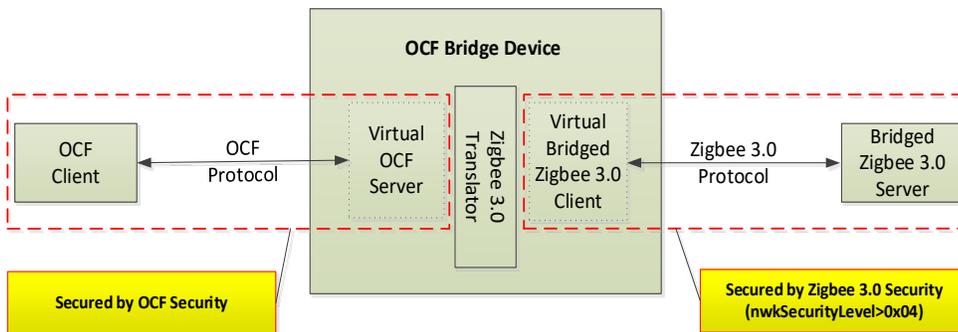
9904 The security level `nwkSecurityLevel > 0x04` provides message integrity code (MIC) and/or AES128-  
 9905 CCM encryption (ENC). Zigbee Servers using `nwkSecurityLevel > 0x04` would typically be  
 9906 considered secure from an OCF perspective. The appropriate selection for `nwkSecurityLevel` is left  
 9907 to the vendor.

9908 See Table E.4 for a summary of the Zigbee Security Levels.

9909 **Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers**

Security Level Identifier	Security Level Sub-Field	Security Attributes	Data Encryption	Frame Integrity (Length of M of MIC, in Number of Octets)
0x00	'000'	None	OFF	NO (M=0)
0x01	'001'	MIC-32	OFF	YES(M=4)
0x02	'010'	MIC-64	OFF	YES(M=8)
0x03	'011'	MIC-128	OFF	YES(M=16)
0x04	'100'	ENC	ON	NO(M=0)
0x05	'101'	ENC-MIC-32	ON	YES(M=4)
0x06	'110'	ENC-MIC-64	ON	YES(M=8)
0x07	'111'	ENC-MIC-128	ON	YES(M=16)

9910 Figure E-3 shows how communications in both ecosystems of OCF Bridge Device are secured by  
 9911 their own security.



9912  
 9913

9914 **Figure E-3 Security Considerations for Zigbee Bridge**