

# OCF Security Specification

VERSION 2.0.2 | April 2019



**OPEN** CONNECTIVITY  
FOUNDATION™

CONTACT [admin@openconnectivity.org](mailto:admin@openconnectivity.org)  
Copyright Open Connectivity Foundation, Inc. © 2019.  
All Rights Reserved.

## 1 **LEGAL DISCLAIMER**

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

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

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

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

# CONTENTS

17			
18	1	Scope.....	1
19	2	Normative References .....	1
20	3	Terms, definitions, and abbreviated terms .....	3
21	3.1	Terms and definitions .....	3
22	3.2	Abbreviated terms .....	6
23	4	Document Conventions and Organization.....	10
24	4.1	Conventions .....	10
25	4.2	Notation .....	10
26	4.3	Data types .....	11
27	4.4	Document structure.....	11
28	5	Security Overview .....	12
29	5.1	Preamble.....	12
30	5.2	Access Control .....	14
31	5.2.1	ACL Architecture .....	16
32	5.2.2	Access Control Scoping Levels .....	19
33	5.3	Onboarding Overview .....	21
34	5.3.1	Onboarding General .....	21
35	5.3.2	Onboarding Steps .....	23
36	5.3.3	Establishing a Device Owner .....	24
37	5.3.4	Provisioning for Normal Operation.....	25
38	5.3.5	Device Provisioning for OCF Cloud and Device Registration Overview.....	25
39	5.3.6	OCF Compliance Management System.....	25
40	5.4	Provisioning.....	26
41	5.4.1	Provisioning General.....	26
42	5.4.2	Provisioning other services .....	26
43	5.4.3	Provisioning Credentials for Normal Operation .....	27
44	5.4.4	Role Assignment and Provisioning for Normal Operation .....	27
45	5.4.5	ACL provisioning .....	27
46	5.5	Secure Resource Manager (SRM).....	27
47	5.6	Credential Overview.....	28
48	6	Security for the Discovery Process.....	29
49	6.1	Preamble.....	29
50	6.2	Security Considerations for Discovery .....	29
51	7	Security Provisioning.....	32
52	7.1	Device Identity.....	32
53	7.1.1	General Device Identity.....	32
54	7.1.2	Device Identity for Devices with UAID [Deprecated] .....	32
55	7.2	Device Ownership.....	32
56	7.3	Device Ownership Transfer Methods.....	33
57	7.3.1	OTM implementation requirements .....	33
58	7.3.2	SharedKey Credential Calculation.....	35
59	7.3.3	Certificate Credential Generation.....	35

60	7.3.4	Just-Works OTM.....	35
61	7.3.5	Random PIN Based OTM .....	37
62	7.3.6	Manufacturer Certificate Based OTM.....	39
63	7.3.7	Vendor Specific OTMs.....	42
64	7.3.8	Establishing Owner Credentials.....	43
65	7.3.9	Security considerations regarding selecting an Ownership Transfer Method..	51
66	7.3.10	Security Profile Assignment.....	51
67	7.4	Provisioning.....	52
68	7.4.1	Provisioning Flows .....	52
69	7.5	Device Provisioning for OCF Cloud .....	57
70	7.5.1	Cloud Provisioning General.....	57
71	7.5.2	Device Provisioning by Mediator.....	57
72	8	Device Onboarding State Definitions .....	58
73	8.1	Device Onboarding General.....	58
74	8.2	Device Onboarding-Reset State Definition.....	60
75	8.3	Device Ready-for-OTM State Definition .....	60
76	8.4	Device Ready-for-Provisioning State Definition .....	61
77	8.5	Device Ready-for-Normal-Operation State Definition.....	61
78	8.6	Device Soft Reset State Definition .....	62
79	9	Security Credential Management.....	65
80	9.1	Preamble.....	65
81	9.2	Credential Lifecycle.....	65
82	9.2.1	Credential Lifecycle General.....	65
83	9.2.2	Creation.....	65
84	9.2.3	Deletion.....	65
85	9.2.4	Refresh .....	65
86	9.2.5	Revocation .....	65
87	9.3	Credential Types.....	66
88	9.3.1	Preamble .....	66
89	9.3.2	Pair-wise Symmetric Key Credentials .....	66
90	9.3.3	Group Symmetric Key Credentials.....	66
91	9.3.4	Asymmetric Authentication Key Credentials.....	67
92	9.3.5	Asymmetric Key Encryption Key Credentials.....	67
93	9.3.6	Certificate Credentials.....	68
94	9.3.7	Password Credentials .....	68
95	9.4	Certificate Based Key Management.....	68
96	9.4.1	Overview.....	68
97	9.4.2	X.509 Digital Certificate Profiles .....	69
98	9.4.3	Certificate Revocation List (CRL) Profile.....	78
99	9.4.4	Resource Model .....	79
100	9.4.5	Certificate Provisioning.....	79
101	9.4.6	CRL Provisioning.....	80
102	10	Device Authentication.....	83
103	10.1	Device Authentication General.....	83

104	10.2	Device Authentication with Symmetric Key Credentials .....	83
105	10.3	Device Authentication with Raw Asymmetric Key Credentials.....	83
106	10.4	Device Authentication with Certificates .....	83
107	10.4.1	Device Authentication with Certificates General .....	83
108	10.4.2	Role Assertion with Certificates .....	84
109	10.4.3	OCF PKI Roots .....	85
110	10.4.4	PKI Trust Store .....	85
111	10.4.5	Path Validation and extension processing .....	86
112	10.5	Device Authentication with OCF Cloud .....	87
113	10.5.1	Device Authentication with OCF Cloud General .....	87
114	10.5.2	Device Connection with the OCF Cloud.....	87
115	10.5.3	Security Considerations .....	88
116	11	Message Integrity and Confidentiality.....	90
117	11.1	Preamble.....	90
118	11.2	Session Protection with DTLS.....	90
119	11.2.1	DTLS Protection General.....	90
120	11.2.2	Unicast Session Semantics .....	90
121	11.2.3	Cloud Session Semantics .....	90
122	11.3	Cipher Suites .....	90
123	11.3.1	Cipher Suites General .....	90
124	11.3.2	Cipher Suites for Device Ownership Transfer.....	90
125	11.3.3	Cipher Suites for Symmetric Keys .....	91
126	11.3.4	Cipher Suites for Asymmetric Credentials .....	92
127	11.3.5	Cipher suites for OCF Cloud Credentials .....	92
128	12	Access Control.....	94
129	12.1	ACL Generation and Management .....	94
130	12.2	ACL Evaluation and Enforcement .....	94
131	12.2.1	ACL Evaluation and Enforcement General .....	94
132	12.2.2	Host Reference Matching .....	94
133	12.2.3	Resource Wildcard Matching .....	94
134	12.2.4	Multiple Criteria Matching.....	95
135	12.2.5	Subject Matching using Wildcards .....	95
136	12.2.6	Subject Matching using Roles .....	95
137	12.2.7	ACL Evaluation .....	96
138	13	Security Resources.....	98
139	13.1	Security Resources General.....	98
140	13.2	Device Owner Transfer Resource .....	100
141	13.2.1	Device Owner Transfer Resource General .....	100
142	13.2.2	Persistent and Semi-Persistent Device Identifiers.....	103
143	13.2.3	Onboarding Considerations for Device Identifier .....	103
144	13.2.4	OCF defined OTMs.....	104
145	13.3	Credential Resource.....	105
146	13.3.1	Credential Resource General.....	105
147	13.3.2	Properties of the Credential Resource .....	109

148	13.3.3	Key Formatting .....	112
149	13.3.4	Credential Refresh Method Details .....	112
150	13.4	Certificate Revocation List.....	114
151	13.4.1	CRL Resource Definition.....	114
152	13.5	ACL Resources.....	114
153	13.5.1	ACL Resources General.....	114
154	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures.....	114
155	13.5.3	ACL Resource .....	115
156	13.6	Access Manager ACL Resource .....	122
157	13.7	Signed ACL Resource.....	123
158	13.8	Provisioning Status Resource.....	124
159	13.9	Certificate Signing Request Resource .....	129
160	13.10	Roles Resource .....	130
161	13.11	Account Resource .....	131
162	13.12	Account Session resource.....	133
163	13.13	Account Token Refresh Resource.....	134
164	13.14	Security Virtual Resources (SVRs) and Access Policy.....	135
165	13.15	SVRs, Discoverability and OCF Endpoints.....	135
166	13.16	Additional Privacy Consideration for Core and SVRs Resources .....	135
167	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ...	135
168	13.16.2	Privacy Protecting the Device Identifiers.....	138
169	13.16.3	Privacy Protecting the Protocol Independent Device Identifier .....	138
170	13.16.4	Privacy Protecting the Platform Identifier.....	139
171	13.17	Easy Setup Resource Device State .....	139
172	14	Security Hardening Guidelines/ Execution Environment Security.....	142
173	14.1	Preamble.....	142
174	14.2	Execution Environment Elements.....	142
175	14.2.1	Execution Environment Elements General.....	142
176	14.2.2	Secure Storage.....	142
177	14.2.3	Secure execution engine.....	145
178	14.2.4	Trusted input/output paths .....	145
179	14.2.5	Secure clock .....	145
180	14.2.6	Approved algorithms .....	145
181	14.2.7	Hardware tamper protection .....	146
182	14.3	Secure Boot.....	146
183	14.3.1	Concept of software module authentication .....	146
184	14.3.2	Secure Boot process.....	148
185	14.3.3	Robustness Requirements .....	148
186	14.4	Attestation .....	148
187	14.5	Software Update .....	148
188	14.5.1	Overview:.....	148
189	14.5.2	Recognition of Current Differences .....	149
190	14.5.3	Software Version Validation .....	149
191	14.5.4	Software Update.....	149

192	14.5.5	Recommended Usage .....	149
193	14.6	Non-OCF Endpoint interoperability .....	150
194	14.7	Security Levels .....	150
195	14.8	Security Profiles .....	151
196	14.8.1	Security Profiles General .....	151
197	14.8.2	Identification of Security Profiles (Normative) .....	151
198	14.8.3	Security Profiles .....	153
199	15	Device Type Specific Requirements .....	158
200	15.1	Bridging Security .....	158
201	15.1.1	Universal Requirements for Bridging to another Ecosystem .....	158
202	15.1.2	Additional Security Requirements specific to Bridged Protocols .....	158
203	Annex A	(informative) Access Control Examples .....	160
204	A.1	Example OCF ACL Resource .....	160
205	A.2	Example AMS .....	160
206	Annex B	(Informative) Execution Environment Security Profiles .....	162
207	Annex C	(normative) Resource Type definitions .....	163
208	C.1	List of Resource Type definitions .....	163
209	C.2	Account Token .....	163
210	C.2.1	Introduction .....	163
211	C.2.2	Well-known URI .....	163
212	C.2.3	Resource type .....	163
213	C.2.4	OpenAPI 2.0 definition .....	163
214	C.2.5	Property definition .....	166
215	C.2.6	CRUDN behaviour .....	167
216	C.3	Access Control List .....	167
217	C.3.1	Introduction .....	167
218	C.3.2	Well-known URI .....	167
219	C.3.3	Resource type .....	167
220	C.3.4	OpenAPI 2.0 definition .....	167
221	C.3.5	Property definition .....	174
222	C.3.6	CRUDN behaviour .....	174
223	C.4	Access Control List-2 .....	174
224	C.4.1	Introduction .....	174
225	C.4.2	Well-known URI .....	175
226	C.4.3	Resource type .....	175
227	C.4.4	OpenAPI 2.0 definition .....	175
228	C.4.5	Property definition .....	184
229	C.4.6	CRUDN behaviour .....	184
230	C.5	Managed Access Control .....	185
231	C.5.1	Introduction .....	185
232	C.5.2	Well-known URI .....	185
233	C.5.3	Resource type .....	185
234	C.5.4	OpenAPI 2.0 definition .....	185
235	C.5.5	Property definition .....	188

236	C.5.6	CRUDN behaviour.....	189
237	C.6	Credential.....	189
238	C.6.1	Introduction.....	189
239	C.6.2	Well-known URI.....	189
240	C.6.3	Resource type.....	189
241	C.6.4	OpenAPI 2.0 definition.....	189
242	C.6.5	Property definition.....	199
243	C.6.6	CRUDN behaviour.....	200
244	C.7	Certificate Revocation.....	200
245	C.7.1	Introduction.....	200
246	C.7.2	Well-known URI.....	200
247	C.7.3	Resource type.....	200
248	C.7.4	OpenAPI 2.0 definition.....	200
249	C.7.5	Property definition.....	202
250	C.7.6	CRUDN behaviour.....	203
251	C.8	Certificate Signing Request.....	203
252	C.8.1	Introduction.....	203
253	C.8.2	Well-known URI.....	203
254	C.8.3	Resource type.....	203
255	C.8.4	OpenAPI 2.0 definition.....	203
256	C.8.5	Property definition.....	205
257	C.8.6	CRUDN behaviour.....	205
258	C.9	Device Owner Transfer Method.....	205
259	C.9.1	Introduction.....	205
260	C.9.2	Well-known URI.....	205
261	C.9.3	Resource type.....	205
262	C.9.4	OpenAPI 2.0 definition.....	205
263	C.9.5	Property definition.....	209
264	C.9.6	CRUDN behaviour.....	211
265	C.10	Device Provisioning Status.....	211
266	C.10.1	Introduction.....	211
267	C.10.2	Well-known URI.....	211
268	C.10.3	Resource type.....	211
269	C.10.4	OpenAPI 2.0 definition.....	211
270	C.10.5	Property definition.....	215
271	C.10.6	CRUDN behaviour.....	219
272	C.11	Asserted Roles.....	220
273	C.11.1	Introduction.....	220
274	C.11.2	Well-known URI.....	220
275	C.11.3	Resource type.....	220
276	C.11.4	OpenAPI 2.0 definition.....	220
277	C.11.5	Property definition.....	229
278	C.11.6	CRUDN behaviour.....	229
279	C.12	Signed Access Control List.....	229



280	C.12.1	Introduction.....	229
281	C.12.2	Well-known URI .....	229
282	C.12.3	Resource type .....	229
283	C.12.4	OpenAPI 2.0 definition.....	229
284	C.12.5	Property definition.....	237
285	C.12.6	CRUDN behaviour.....	237
286	C.13	Session.....	238
287	C.13.1	Introduction.....	238
288	C.13.2	Well-known URI .....	238
289	C.13.3	Resource type .....	238
290	C.13.4	OpenAPI 2.0 definition.....	238
291	C.13.5	Property definition.....	240
292	C.13.6	CRUDN behaviour.....	240
293	C.14	Security Profile .....	241
294	C.14.1	Introduction.....	241
295	C.14.2	Well-known URI .....	241
296	C.14.3	Resource type .....	241
297	C.14.4	OpenAPI 2.0 definition.....	241
298	C.14.5	Property definition.....	243
299	C.14.6	CRUDN behaviour.....	243
300	C.15	Token Refresh.....	244
301	C.15.1	Introduction.....	244
302	C.15.2	Well-known URI .....	244
303	C.15.3	Resource type .....	244
304	C.15.4	OpenAPI 2.0 definition.....	244
305	C.15.5	Property definition.....	246
306	C.15.6	CRUDN behaviour.....	247
307	Annex D (informative)	OID definitions .....	248
308	Annex E (informative)	Security considerations specific to Bridged Protocols.....	250
309	E.1	Security Considerations specific to the AllJoyn Protocol.....	250
310	E.2	Security Considerations specific to the Bluetooth LE Protocol.....	250
311	E.3	Security Considerations specific to the oneM2M Protocol .....	250
312	E.4	Security Considerations specific to the U+ Protocol.....	251
313	E.5	Security Considerations specific to the Z-Wave Protocol .....	251
314	E.6	Security Considerations specific to the Zigbee Protocol .....	252
315			

316 **FIGURES**

317 Figure 1 – OCF Interaction..... 10

318 Figure 2 – OCF Layers ..... 12

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

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

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

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

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

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

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

326 Figure 10 – Onboarding Overview ..... 22

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

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

329 Figure 13 – Discover New Device Sequence..... 34

330 Figure 14 – A Just Works OTM..... 36

331 Figure 15 – Random PIN-based OTM..... 37

332 Figure 16 – Manufacturer Certificate Based OTM Sequence ..... 41

333 Figure 17 – Vendor-specific Owner Transfer Sequence ..... 42

334 Figure 18 – Establish Device Identity Flow..... 44

335 Figure 19 – Owner Credential Selection Provisioning Sequence..... 45

336 Figure 20 – Symmetric Owner Credential Provisioning Sequence ..... 46

337 Figure 21 – Asymmetric Owner Credential Provisioning Sequence..... 47

338 Figure 22 – Configure Device Services ..... 49

339 Figure 23 – Provision New Device for Peer to Peer Interaction Sequence ..... 50

340 Figure 24 – Example of Client-directed provisioning..... 53

341 Figure 25 – Example of Server-directed provisioning using a single provisioning service..... 54

342 Figure 26 – Example of Server-directed provisioning involving multiple support services..... 57

343 Figure 27 – Device state model ..... 59

344 Figure 28 – OBT Sanity Check Sequence in SRESET..... 63

345 Figure 29 – Client-directed Certificate Transfer..... 80

346 Figure 30 – Client-directed CRL Transfer..... 81

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

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

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

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

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

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

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

354 Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states..... 139

355	Figure 40 – Software Module Authentication.....	147
356	Figure 41 – Verification Software Module.....	147
357	Figure 42 – Software Module Authenticity .....	148
358	Figure A-1 – Example "/oic/sec/acl2" Resource.....	160
359	Figure A-2 Example "/oic/sec/amacl" Resource.....	161
360	Figure E-1 Security Considerations for BLE Bridge .....	250
361	Figure E-2 Security Considerations for Z-Wave Bridge.....	252
362	Figure E-3 Security Considerations for Zigbee Bridge.....	253
363		

364 **Tables**

365 Table 1 – Discover New Device Details ..... 34

366 Table 2 – A Just Works OTM Details ..... 36

367 Table 3 – Random PIN-based OTM Details ..... 38

368 Table 4 – Manufacturer Certificate Based OTM Details ..... 41

369 Table 5 – Vendor-specific Owner Transfer Details ..... 43

370 Table 6 – Establish Device Identity Details ..... 44

371 Table 7 – Owner Credential Selection Details ..... 45

372 Table 8 – Symmetric Owner Credential Assignment Details ..... 46

373 Table 9 – Asymmetric Owner Credential Assignment Details ..... 47

374 Table 10 – Configure Device Services Detail ..... 49

375 Table 11 – Provision New Device for Peer to Peer Details ..... 50

376 Table 12 – Steps describing Client -directed provisioning ..... 53

377 Table 13 – Steps for Server-directed provisioning using a single provisioning service ..... 54

378 Table 14 – Steps for Server-directed provisioning involving multiple support services ..... 57

379 Table 15 – Mapping of Properties of the "oic.r.account" and "oic.r.coapcloudconf"  
380 Resources ..... 58

381 Table 16 – X.509 v1 fields for Root CA Certificates ..... 69

382 Table 17 - X.509 v3 extensions for Root CA Certificates ..... 70

383 Table 18 - X.509 v1 fields for Intermediate CA Certificates ..... 70

384 Table 19 – X.509 v3 extensions for Intermediate CA Certificates ..... 71

385 Table 20 – X.509 v1 fields for End-Entity Certificates ..... 71

386 Table 21 – X.509 v3 extensions for End-Entity Certificates ..... 72

387 Table 22 – Device connection with the OCF Cloud flow ..... 88

388 Table 23 – ACE2 Wildcard Matching Strings Description ..... 94

389 Table 24 – Definition of the "/oic/sec/doxm" Resource ..... 101

390 Table 25 – Properties of the "/oic/sec/doxm" Resource ..... 101

391 Table 26 – Properties of the "/oic/sec/didtype" Property ..... 102

392 Table 27 – Properties of the "oic.sec.doxmtype" Property ..... 104

393 Table 28 – Definition of the "oic.r.cred" Resource ..... 105

394 Table 29 – Properties of the "/oic/sec/cred" Resource ..... 106

395 Table 30 – Properties of the "oic.sec.cred" Property ..... 107

396 Table 31: Properties of the "oic.sec.credusagetype" Property ..... 108

397 Table 32 – Properties of the "oic.sec.pubdatatype" Property ..... 108

398 Table 33 – Properties of the "oic.sec.privdatatype" Property ..... 108

399 Table 34 – Properties of the "oic.sec.optdatatype" Property ..... 109

400 Table 35 – Definition of the "oic.sec.roletype" Property ..... 109

401 Table 36 – Value Definition of the "oic.sec.crmttype" Property ..... 111

402 Table 37 – 128-bit symmetric key ..... 112

403	Table 38 – 256-bit symmetric key .....	112
404	Table 39 – Definition of the "oic.r.crl" Resource .....	114
405	Table 40 – Properties of the "oic.r.crl" Resource .....	114
406	Table 41 – BNF Definition of OCF ACL .....	114
407	Table 42 – Definition of the "oic.r.acl" Resource.....	117
408	Table 43 – Properties of the "oic.r.acl" Resource.....	117
409	Table 44 – Properties of the "oic.r.ace" Property .....	118
410	Table 45 – Value Definition of the "oic.sec.crudtype" Property .....	118
411	Table 46 – Definition of the "oic.sec.acl2" Resource.....	119
412	Table 47 – Properties of the "oic.sec.acl2" Resource.....	119
413	Table 48 – "oic.sec.ace2" data type definition.....	120
414	Table 49 – "oic.sec.ace2.resource-ref" data type definition.....	120
415	Table 50 – Value definition "oic.sec.conntype" Property .....	120
416	Table 51 – Definition of the "oic.r.amacl" Resource.....	123
417	Table 52 – Properties of the "oic.r.amacl" Resource.....	123
418	Table 53 – Definition of the "oic.r.sacl" Resource .....	123
419	Table 54 – Properties of the "oic.r.sacl" Resource .....	123
420	Table 55 – Properties of the "oic.sec.sigtype" Property .....	124
421	Table 56 – Definition of the "oic.r.pstat" Resource .....	124
422	Table 57 – Properties of the "oic.r.pstat" Resource .....	125
423	Table 58 – Properties of the "/oic/sec/dostype" Property.....	126
424	Table 59 – Definition of the "oic.sec.dpmttype" Property.....	128
425	Table 60 – Value Definition of the "oic.sec.dpmttype" Property (Low-Byte) .....	128
426	Table 61 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte).....	128
427	Table 62 – Definition of the "oic.sec.pomttype" Property.....	129
428	Table 63 – Value Definition of the "oic.sec.pomttype" Property .....	129
429	Table 64 – Definition of the "oic.r.csr" Resource .....	129
430	Table 65 – Properties of the "oic.r.csr" Resource .....	129
431	Table 66 – Definition of the "oic.r.roles" Resource .....	131
432	Table 67 – Properties of the "oic.r.roles" Resource .....	131
433	Table 68 – Definition of the "oic.r.account" Resource.....	132
434	Table 69 – Properties of the "oic.r.account" Resource.....	132
435	Table 70 – Definition of the "oic.r.session" Resource.....	133
436	Table 71 – Properties of the "oic.r.session" Resource.....	133
437	Table 72 – Definition of the "oic.r.tokenrefresh" Resource .....	134
438	Table 73 – Properties of the "oic.r.tokenrefresh" Resource .....	135
439	Table 74 – Core Resource Properties Access Modes given various Device States .....	137
440	Table 75 – Examples of Sensitive Data .....	143
441	Table 76 – Definition of the "oic.sec.sp" Resource .....	152

442	Table 77 – Properties of the "oic.sec.sp" Resource .....	152
443	Table B.1 – OCF Security Profile .....	162
444	Table C.1 – Alphabetized list of security resources .....	163
445	Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account".....	166
446	Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account". .....	167
447	Table C.4 – The Property definitions of the Resource with type "rt" = "oic.r.acl".....	174
448	Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl". .....	174
449	Table C.6 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".....	184
450	Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".....	184
451	Table C.8 – The Property definitions of the Resource with type "rt" = "oic.r.amacl".....	188
452	Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl". .....	189
453	Table C.10 – The Property definitions of the Resource with type "rt" = "oic.r.cred". .....	199
454	Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".....	200
455	Table C.12 – The Property definitions of the Resource with type "rt" = "oic.r.crl".....	202
456	Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl". .....	203
457	Table C.14 – The Property definitions of the Resource with type "rt" = "oic.r.csr".....	205
458	Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".....	205
459	Table C.16 – The Property definitions of the Resource with type "rt" = "oic.r.doxm". .....	209
460	Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxm".....	211
461	Table C.18 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".....	215
462	Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat". .....	219
463	Table C.20 – The Property definitions of the Resource with type "rt" = "oic.r.roles".....	229
464	Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles". .....	229
465	Table C.22 – The Property definitions of the Resource with type "rt" = "oic.r.sacl".....	237
466	Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl". .....	238
467	Table C.24 – The Property definitions of the Resource with type "rt" = "oic.r.session".....	240
468	Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.session".....	240
469	Table C.26 – The Property definitions of the Resource with type "rt" = "oic.r.sp".....	243
470	Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp". .....	243
471	Table C.28 – The Property definitions of the Resource with type "rt" = "oic.r.tokenrefresh".	246
472	Table C.29 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh".	247
473	Table E.1 GAP security mode .....	250
474	Table E.2 TLS 1.2 Cipher Suites used by U+ .....	251
475	Table E.3 Z-Wave Security Class.....	252
476	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers.....	253
477		
478		

484 **1 Scope**

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

489 **2 Normative References**

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

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

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

504 OCF Wi-Fi Easy Setup Specification  
505 Latest version available at:  
506 [https://openconnectivity.org/specs/OCF\\_Wi-Fi\\_Easy\\_Setup\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf)

507 OCF Cloud Specification  
508 Latest version available at:  
509 [https://openconnectivity.org/specs/OCF\\_Cloud\\_Specification.pdf](https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf)

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

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

513 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September  
514 2000, <https://tools.ietf.org/html/rfc2898>

515 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November  
516 2000, <https://tools.ietf.org/html/rfc2986>

517 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December  
518 2005, <https://tools.ietf.org/html/rfc4279>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

553



554 **3 Terms, definitions, and abbreviated terms**

555 **3.1 Terms and definitions**

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

558 ISO and IEC maintain terminological databases for use in standardization at the following  
559 addresses:

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

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

562 **3.1.1**

563 **Access Management Service (AMS)**

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

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

567 **3.1.2**

568 **Access Token**

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

571 **3.1.3**

572 **Authorization Provider**

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

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

576 **3.1.4**

577 **Client**

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

579 **3.1.5**

580 **Credential Management Service (CMS)**

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

583 **3.1.6**

584 **Device**

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

586 **3.1.7**

587 **Device Class**

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

591 **3.1.8**

592 **Device ID**

593 a stack instance identifier.

594 **3.1.9**

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

596 a logical entity that establishes device ownership

597 **3.1.10**  
598 **Device Registration**  
599 a process by which Device is enrolled/registered to the OCF Cloud infrastructure (using Device  
600 certificate and unique credential) and becomes ready for further remote operation through the  
601 cloud interface (e.g. connection to remote Resources or publishing of its own Resources for  
602 access).

603 **3.1.11**  
604 **End-Entity**  
605 any certificate holder which is not a Root or Intermediate Certificate Authority.

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

607 **3.1.12**  
608 **Entity**  
609 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

610 **3.1.13**  
611 **OCF Interface**  
612 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

613 **3.1.14**  
614 **Intermediary**  
615 a Device that implements both Client and Server roles and may perform protocol translation,  
616 virtual device to physical device mapping or Resource translation

617 **3.1.15**  
618 **OCF Cipher Suite**  
619 a set of algorithms and parameters that define the cryptographic functionality of a Device. The  
620 OCF Cipher Suite includes the definition of the public key group operations, signatures, and  
621 specific hashing and encoding used to support the public key.

622 **3.1.16**  
623 **OCF Cloud User**  
624 a person or organization authorizing a set of Devices to interact with each other via an OCF  
625 Cloud.

626 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  
627 organization that onboarded that Device. The OCF Cloud User delegates, to the OCF Cloud authority, authority to route  
628 between Devices registered by the OCF Cloud User. The OCF Cloud delegates, to the OCF Cloud User, authority to  
629 select the set of Devices which can register and use the services of the OCF Cloud.

630 **3.1.17**  
631 **OCF Rooted Certificate Chain**  
632 a collection of X.509 v3 certificates in which each certificate chains to a trust anchor certificate  
633 which has been issued by a certificate authority under the direction, authority, and approval of  
634 the Open Connectivity Foundation Board of Directors as a trusted root for the OCF ecosystem.

635 **3.1.18**  
636 **Onboarding Tool (OBT)**  
637 a tool that implements DOTS(3.1.9), AMS(3.1.1) and CMS(3.1.5) functionality

638 **3.1.19**  
639 **Out of Band Method**  
640 any mechanism for delivery of a secret from one party to another, not specified by OCF

641 **3.1.20**  
642 **Owner Credential (OC)**  
643 credential, provisioned by an OBT(3.1.18) to a Device during onboarding, for the purposes of  
644 mutual authentication of the Device and OBT(3.1.18) during subsequent interactions

645 **3.1.21**  
646 **Platform ID**  
647 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

648 **3.1.22**  
649 **Property**  
650 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

651 **3.1.23**  
652 **Resource**  
653 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

654 **3.1.24**  
655 **Role (Network context)**  
656 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

657 **3.1.25**  
658 **Role Identifier**  
659 a Property of an OCF credentials Resource or element in a role certificate that identifies a  
660 privileged role that a Server Device associates with a Client Device for the purposes of making  
661 authorization decisions when the Client Device requests access to Device Resources.

662 **3.1.26**  
663 **Secure Resource Manager (SRM)**  
664 a module in the OCF Core that implements security functionality that includes management of  
665 security Resources such as ACLs, credentials and Device owner transfer state.

666 **3.1.27**  
667 **Security Virtual Resource (SVR)**  
668 a resource supporting security features.

669 Note 1 to entry: For a list of all the SVRs please see clause 13.

670 **3.1.28**  
671 **Server**  
672 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

673 **3.1.29**  
674 **Trust Anchor**  
675 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.  
676 a Device and an OBT(3.1.18)) can assume trust

677 **3.1.30**  
678 **Unique Authenticable Identifier**  
679 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is  
680 used to create the Device ID.

681 Note 1 to entry: The ownership of a UAID may be authenticated by peer Devices.

682 **3.1.31**  
683 **Device Configuration Resource (DCR)**  
684 a Resource that is any of the following:

685 a) a Discovery Core Resource, or  
686 b) a Security Virtual Resource, or  
687 c) a Wi-Fi Easy Setup Resource, or  
688 d) a CoAP Cloud Configuration Resource.

689 **3.1.32**  
690 **Non-Configuration Resource (NCR)**  
691 a Resource that is not a Device Configuration Resource (3.1.31).

692 **3.1.33**  
693 **Bridged Device**  
694 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

695 **3.1.34**  
696 **Bridged Protocol**  
697 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

698 **3.1.35**  
699 **Bridge**  
700 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

701 **3.1.36**  
702 **Bridging Platform**  
703 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

704 **3.1.37**  
705 **Virtual Bridged Device**  
706 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

707 **3.1.38**  
708 **Virtual OCF Device**  
709 Note 1 to entry: The details are defined in ISO/IEC 30118-3:2018.

710 **3.1.39**  
711 **OCF Security Domain**  
712 set of onboarded OCF Devices that are provisioned with credentialing information for confidential  
713 communication with one another

714 **3.1.40**  
715 **Owned (or "in Owned State")**  
716 having the "owned" Property of the "/oic/sec/doxm" resource equal to "TRUE"

717 **3.1.41**  
718 **Unowned (or "in Unowned State")**  
719 having the "owned" Property of the "/oic/sec/doxm" resource equal to "FALSE"

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 JavaScript Object Notation.

776 Note 1 to entry: See ISO/IEC 30118-1:2018.

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

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

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

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

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

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

793 **3.2.24**  
794 **OC**  
795 Owner Credential

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

799 **3.2.26**  
800 **OBT**  
801 Onboarding Tool

802 **3.2.27**  
803 **OID**  
804 Object Identifier

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

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

811 **3.2.30**  
812 **OWASP**  
813 Open Web Application Security Project.

814 Note 1 to entry: See <https://www.owasp.org/>

815 **3.2.31**  
816 **PE**  
817 Policy Engine

818 **3.2.32**  
819 **PIN**  
820 Personal Identification Number

821 **3.2.33**  
822 **PPSK**  
823 PIN-authenticated pre-shared key

824 **3.2.34**  
825 **PRF**  
826 Pseudo Random Function

827 **3.2.35**  
828 **PSI**  
829 Persistent Storage Interface

830 **3.2.36**  
831 **PSK**  
832 Pre Shared Key

833 **3.2.37**  
834 **RBAC**  
835 Role Based Access Control

836 **3.2.38**  
837 **RM**  
838 Resource Manager

839 **3.2.39**  
840 **RNG**  
841 Random Number Generator

842 **3.2.40**  
843 **SACL**  
844 Signed Access Control List

845 **3.2.41**  
846 **SBAC**  
847 Subject Based Access Control

848 **3.2.42**  
849 **SEE**  
850 Secure Execution Environment

851 **3.2.43**  
852 **SRM**  
853 Secure Resource Manager

854 **3.2.44**  
855 **SVR**  
856 Security Virtual Resource

857 **3.2.45**  
858 **SW**  
859 Software

860 **3.2.46**  
861 **UAID**  
862 Unique Authenticable Identifier

863 **3.2.47**  
864 **URI**  
865 Uniform Resource Identifier

866 Note 1 to entry: See ISO/IEC 30118-1:2018.

867 **3.2.48**  
868 **VOD**  
869 Virtual OCF Device

870 Note 1 to entry: See ISO/IEC 30118-3:2018.

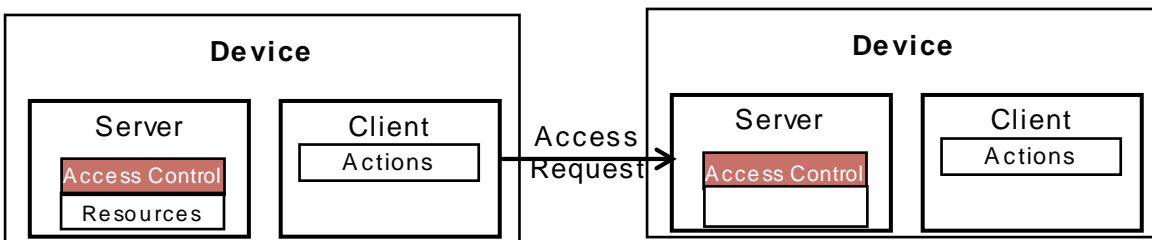
## 871 **4 Document Conventions and Organization**

### 872 **4.1 Conventions**

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

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

877 Figure 1 depicts interaction between OCF Devices.



878

879 **Figure 1 – OCF Interaction**

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

### 884 **4.2 Notation**

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

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

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

891 **Recommended (or should).**

892 These features add functionality supported by OCF Core Architecture and should be implemented.  
893 Recommended features take advantage of the capabilities OCF Core Architecture, usually  
894 without imposing major increase of complexity. Notice that for compliance testing, if a  
895 recommended feature is implemented, it shall meet the specified requirements to be in



896 compliance with these guidelines. Some recommended features could become requirements in  
897 the future. The phrase "should not" indicates behaviour that is permitted but not recommended.

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

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

902 **Conditionally allowed** (CA)

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

905 **Conditionally required** (CR)

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

909 **DEPRECATED**

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

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

917 Words that are emphasized are printed in italic.

### 918 **4.3 Data types**

919 See ISO/IEC 30118-1:2018.

### 920 **4.4 Document structure**

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

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

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

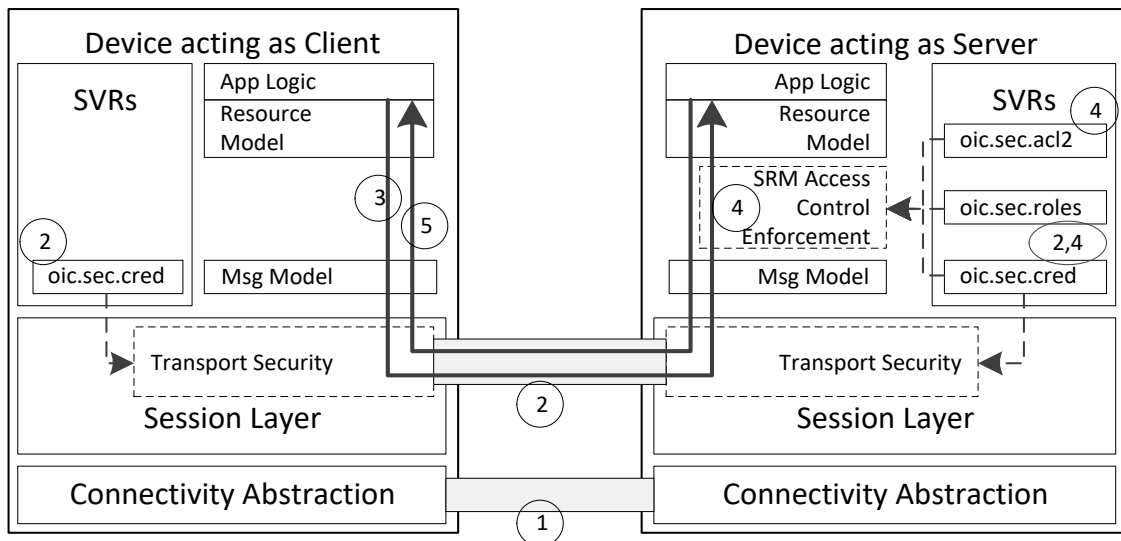
927

928 **5 Security Overview**

929 **5.1 Preamble**

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

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



939  
940

941 **Figure 2 – OCF Layers**

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

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

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

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

960 4) The Server receives the request.

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

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

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

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

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

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

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

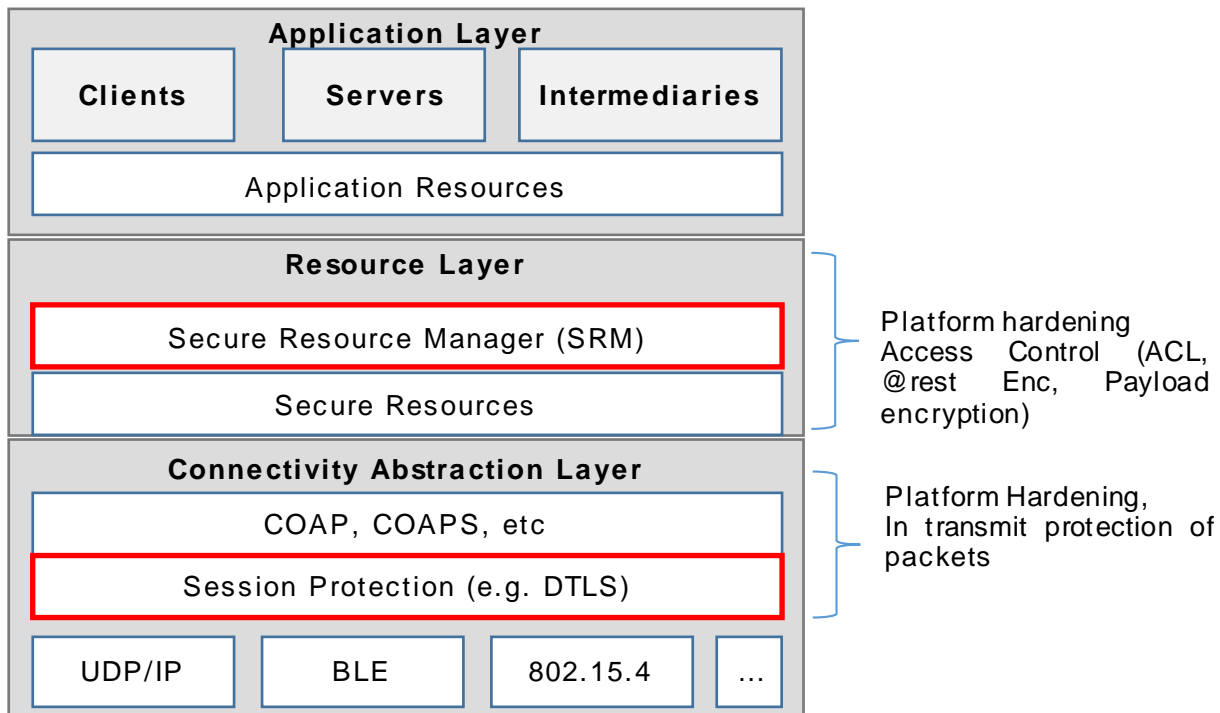
978 5) The Server sends a response back to the Client.

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

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

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

993 Figure 3 depicts OCF Security Enforcement Points.



994  
995  
996 **Figure 3 – OCF Security Enforcement Points**

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

- 999 – Device and Mediator connect over DTLS using "/oic/sec/cred"  
1000 – Device is provisioned by Mediator with following information:  
1001 – the URI of OCF Cloud  
1002 – Token that can be validated by the OCF Cloud  
1003 – UUID of the OCF Cloud

1004 The OpenAPI 2.0 definitions (Annex C) used in this document are normative. This includes that  
1005 all defined payloads shall comply with the indicated OpenAPI 2.0 definitions. Annex C contains all  
1006 of the OpenAPI 2.0 definitions for Resource Types defined in this document.

1007 **5.2 Access Control**

1008 The OCF framework assumes that Resources are hosted by a Server and are made available to  
1009 Clients subject to access control and authorization mechanisms. The Resources at the end point  
1010 are protected through implementation of access control, authentication and confidentiality  
1011 protection. This clause provides an overview of Access Control (AC) through the use of ACLs.  
1012 However, AC in the OCF stack is expected to be transport and connectivity abstraction layer  
1013 agnostic.

1014 Implementation of access control relies on a-priori definition of a set of access policies for the  
1015 Resource. The policies may be stored by a local ACL or an Access Management Service (AMS)  
1016 in form of Access Control Entries (ACE). Two types of access control mechanisms can be applied:

- 1017 – Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of  
1018 requestor) of the requesting entity against the subject included in the policy defined for  
1019 Resource. Asserting the identity of the requestor requires an authentication process.

1020 – Role-based Access Control (RBAC), where each ACE will match a role identifier included in  
1021 the policy for the Resource to a role identifier associated with the requestor

1022 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only  
1023 local references and there is an ACE matching the Atomic Measurement Resource which permits  
1024 the request, then the corresponding requests to linked Resources are permitted by the OCF  
1025 Server. The present paragraph shall apply to any Resource Type based on the Atomic  
1026 Measurement Resource Type.

1027 NOTE The definition of an Atomic Measurement Resource prohibits direct access to the linked Resources. The nature  
1028 of an Atomic Measurement also prohibits updating the "links" to add or remove Resources. Consequently, there is no  
1029 risk of privilege escalation when using the ACE of an Atomic Measurement Resource to govern access to its linked  
1030 Resources.

1031 If an OCF Server receives a batch request to a Collection Resource containing only local  
1032 references and there is an ACE matching the Collection Resource which permits the request,  
1033 then the corresponding requests to linked Resources are permitted by the OCF Server. The  
1034 present paragraph shall apply to any Resource Type based on the Collection Resource Type.

1035 NOTE This implies that the ACEs of the Collection Resource permit access to all the Collection's linked Resources via  
1036 the batch OCF Interface, even if there are no ACEs permitting direct access to some or all the linked Resources. If not  
1037 tightly governed, this could lead to privilege escalation. Restrictions on the use of Collection Resources have been  
1038 provided in ISO/IEC 30118-1:2018 to mitigate the risk of privilege escalation. For example, ISO/IEC 30118-1:2018  
1039 prohibits updating "links" of a Collection Resource with the intent of obtaining access to the added Resource according  
1040 to the ACEs of the Collection, when access to the Resource would have otherwise been denied.

1041 In the OCF access control model, access to a Resource instance requires an associated access  
1042 control policy. This means, each Device acting as Server, needs to have an ACE permitting  
1043 access to each Resource it is protecting. This criterion can be satisfied for a Resource A if there  
1044 is an ACE permitting batch requests to access Resource B containing a Link to Resource A, even  
1045 if there are no ACEs permitting requests which access Resource A directly. Examples of the  
1046 Resource Type for Resource B is the Atomic Measurement Resource Type and the Collection  
1047 Resource Type. The lack of an ACE permitting access to a Resource, either directly or via a Link  
1048 results in the Resource being inaccessible.

1049 The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested  
1050 Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier  
1051 or (3) wildcard. The way in which the client connects to the server may be relevant context for  
1052 making access control decisions. Wildcard matching on authenticated vs. unauthenticated and  
1053 encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject  
1054 classes.

1055 Example Wildcard Matching Policy:

```
1056 "aclist2": [  
1057   {  
1058     "subject": {"conntype" : "anon-clear" },  
1059     "resources": [  
1060       { "wc": "*" }  
1061     ],  
1062     "permission": 31  
1063   },  
1064   {  
1065     "subject": {"conntype" : "auth-crypt" },  
1066     "resources": [  
1067       { "wc": "*" }  
1068     ],
```

1069 "permission": 31  
1070 },  
1071 ]

1072 Details of the format for ACL are defined in clause 12. The ACL is composed of one or more  
1073 ACEs. The ACL defines the access control policy for the Devices.

1074 ACL Resource requires the same security protection as other sensitive Resources, when it comes  
1075 to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW  
1076 and SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may  
1077 have different scoping levels and thus hardening needs to be specially considered for each  
1078 scoping level. For instance, a physical device may host multiple Device implementations and thus  
1079 secure storage, usage and isolation of ACLs for different Servers on the same Device needs to  
1080 be considered.

## 1081 **5.2.1 ACL Architecture**

### 1082 **5.2.1.1 ACL Architecture General**

1083 The Server examines the Resource(s) requested by the client before processing the request. The  
1084 access control resources (e.g. "/oic/sec/acl", "/oic/sec/acl2") are searched to find one or more  
1085 ACE entries that match the requestor and the requested Resources. If a match is found, then  
1086 permission and period constraints are applied. If more than one match is found, then the logical  
1087 UNION of permissions is applied to the overlapping periods.

1088 The server uses the connection context to determine whether the subject has authenticated or  
1089 not and whether data confidentiality has been applied or not. Subject matching wildcard policies  
1090 can match on each aspect. If the user has authenticated, then subject matching may happen at  
1091 increased granularity based on role or device identity.

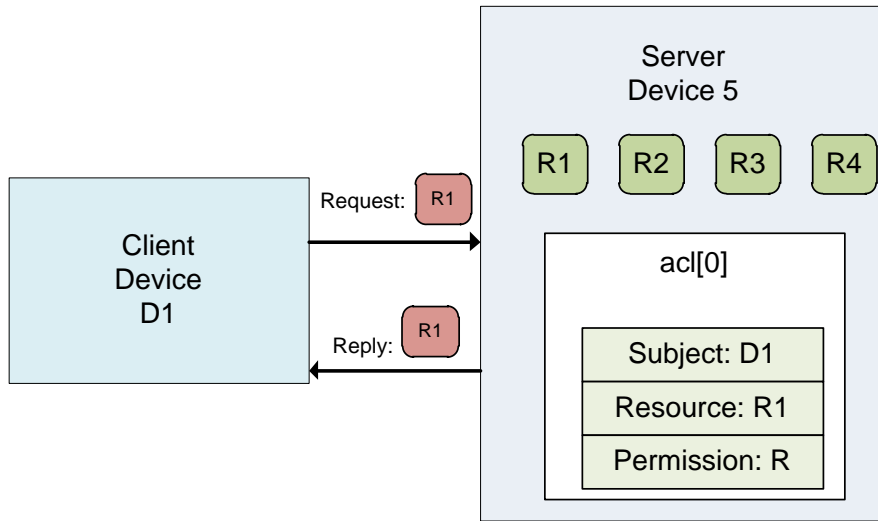
1092 Each ACE contains the permission set that will be applied for a given Resource requestor.  
1093 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY  
1094 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more  
1095 roles. Devices may acquire elevated access permissions when asserting a role. For example, an  
1096 ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally  
1097 accessible.

### 1098 **5.2.1.2 Use of local ACLs**

1099 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control  
1100 processing than remote ACL processing by an AMS.

1101 The following use cases describe the operation of access control

1102 Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4).  
1103 Client Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0]  
1104 corresponds to Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives  
1105 access to Resource R1 because the local ACL "/oic/sec/acl/0" matches the request.

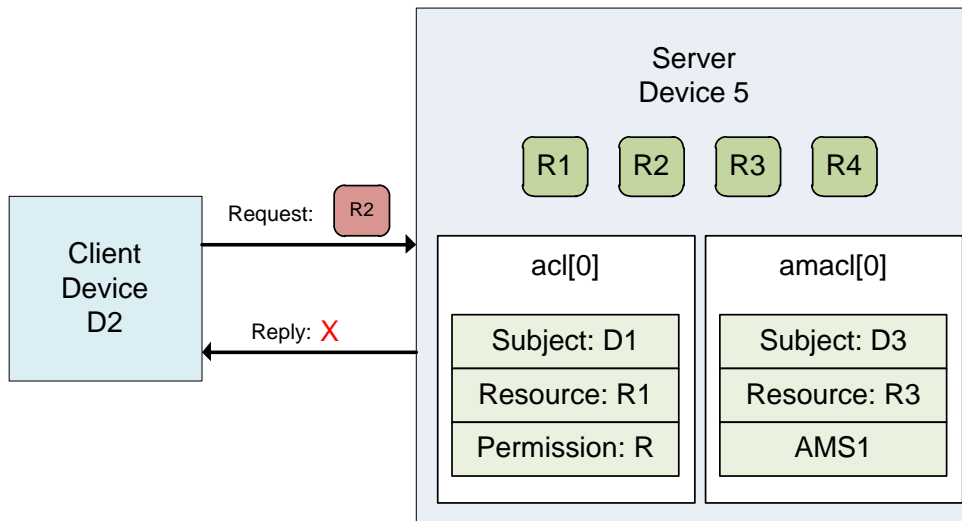


1106

1107

**Figure 4 – Use case-1 showing simple ACL enforcement**

1108 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL  
 1109 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.



1110

1111

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

1112 **5.2.1.3 Use of AMS**

1113 AMS improves ACL policy management. However, they can become a central point of failure.  
 1114 Due to network latency overhead, ACL processing may be slower through an AMS.

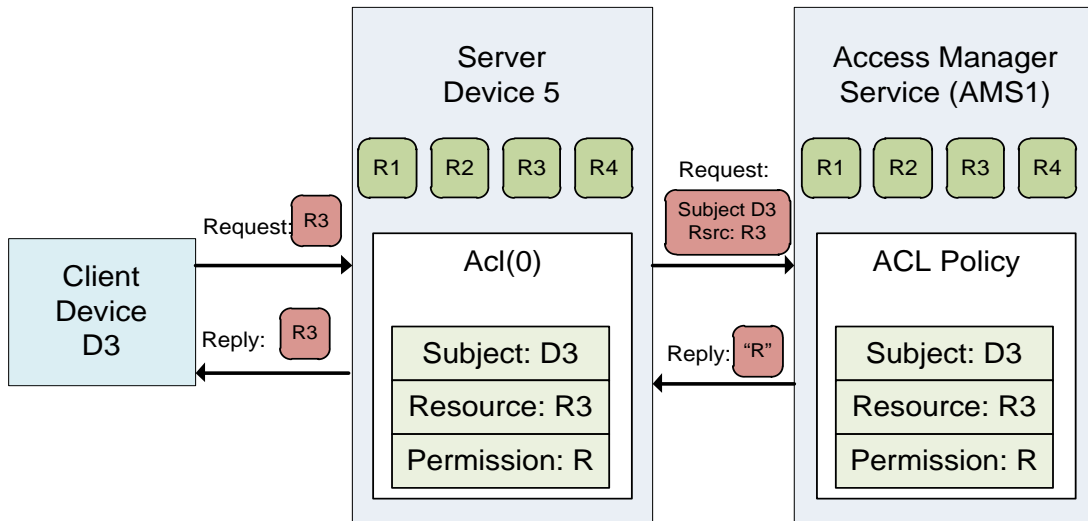
1115 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The  
 1116 Server shall determine which ACL mechanism to use for which Resource set. The  
 1117 "/oic/sec/amacl" Resource is an ACL structure that specifies which Resources will use an AMS to  
 1118 resolve access decisions. The "/oic/sec/amacl" may be used in concert with local ACLs  
 1119 ("/oic/sec/acl").

1120 The AMS is authenticated by referencing a credential issued to the device identifier contained in  
 1121 "/oic/sec/acl2.rowneruuid".

1122 The Server Device may proactively open a connection to the AMS using the Device ID found in  
 1123 "/oic/sec/acl2.rowneruuid". Alternatively, the Server may reject the Resource access request with  
 1124 an error, ACCESS\_DENIED\_REQUIRES\_SACL that instructs the requestor to obtain a suitable  
 1125 ACE policy using a SACL Resource "/oic/sec/sacl". The "/oic/sec/sacl" signature may be  
 1126 validated using the credential Resource associated with the "/oic/sec/acl2.rowneruuid".

1127 The following use cases describe access control using the AMS:

1128 Use Case 3: As depicted in Figure 6, Device D3 requests and receives access to Resource R3  
 1129 with permission Perm1 because the "/oic/sec/amacl/0" matches a policy to consult the Access  
 1130 Manager Server AMS1 service



1131

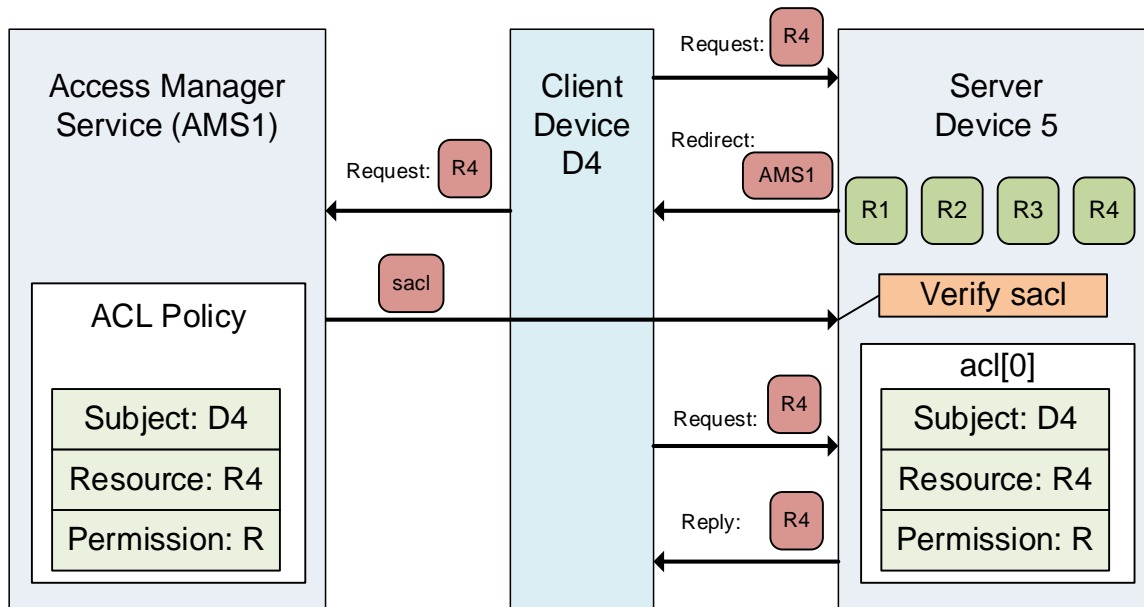
1132 **Figure 6 – Use case-3 showing AMS supported ACL**

1133 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from  
 1134 Server Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1  
 1135 by returning an error identifying AMS1 as a "/oic/sec/sacl" Resource issuer. Device D4 obtains  
 1136 Sacl1 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the  
 1137 "/oic/sec/sacl" Resource and evaluates the ACE policy that grants Perm2 access.

1138 ACE redirection may occur when D4 receives an error result with reason code indicating no  
 1139 match exists (i.e. ACCESS\_DENIED\_NO\_ANCE). D4 reads the "/oic/sec/acl2" Resource to find the  
 1140 "rowneruuid" which identifies the AMS and then submits a request to be provisioned, in this  
 1141 example the AMS chooses to supply a SACL Resource, however it may choose to re-provision  
 1142 the local ACL Resources "/oic/sec/acl" and "/oic/sec/acl2". The request is reissued subsequently.



1143 D4 is presumed to have been introduced to the AMS as part of Device onboarding or through  
 1144 subsequent credential provisioning actions.  
 1145 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.



1146  
 1147 **Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS**

1148 **5.2.2 Access Control Scoping Levels**

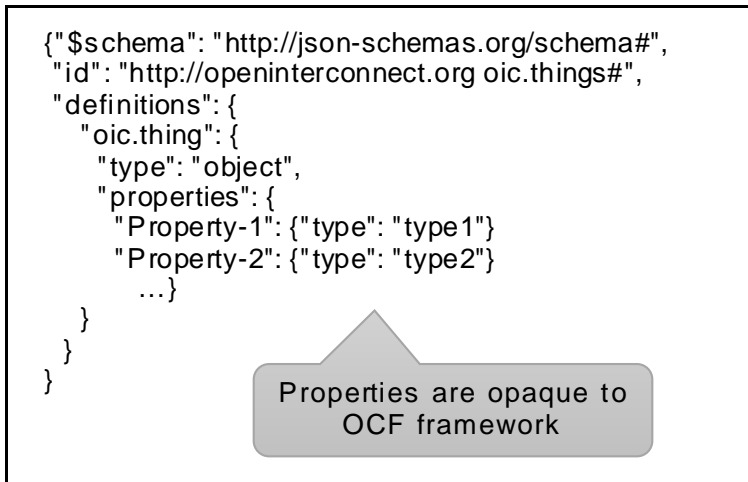
1149 **Group Level Access** - Group scope means applying AC to the group of Devices that are grouped  
 1150 for a specific context. Group Level Access means all group members have access to group data  
 1151 but non-group members must be granted explicit access. Group level access is implemented  
 1152 using Role Credentials and/or connection type

1153 **OCF Device Level Access** – OCF Device scope means applying AC to an individual Device,  
 1154 which may contain multiple Resources. Device level access implies accessibility extends to all  
 1155 Resources available to the Device identified by Device ID. Credentials used for AC mechanisms  
 1156 at Device are OCF Device-specific.

1157 **OCF Resource Level Access** – OCF Resource level scope means applying AC to individual  
 1158 Resources. Resource access requires an ACL that specifies how the entity holding the Resource  
 1159 (Server) shall make a decision on allowing a requesting entity (Client) to access the Resource.

1160 **Property Level Access** - Property level scope means applying AC only to an individual Property.  
 1161 Property level access control is only achieved by creating a Resource that contains a single  
 1162 Property.

1163 Controlling access to static Resources where it is impractical to redesign the Resource, it may  
 1164 appropriate to introduce a collection Resource that references the child Resources having  
 1165 separate access permissions. An example is shown Figure 8, where an "oic.thing" Resource has  
 1166 two properties: Property-1 and Property-2 that would require different permissions.

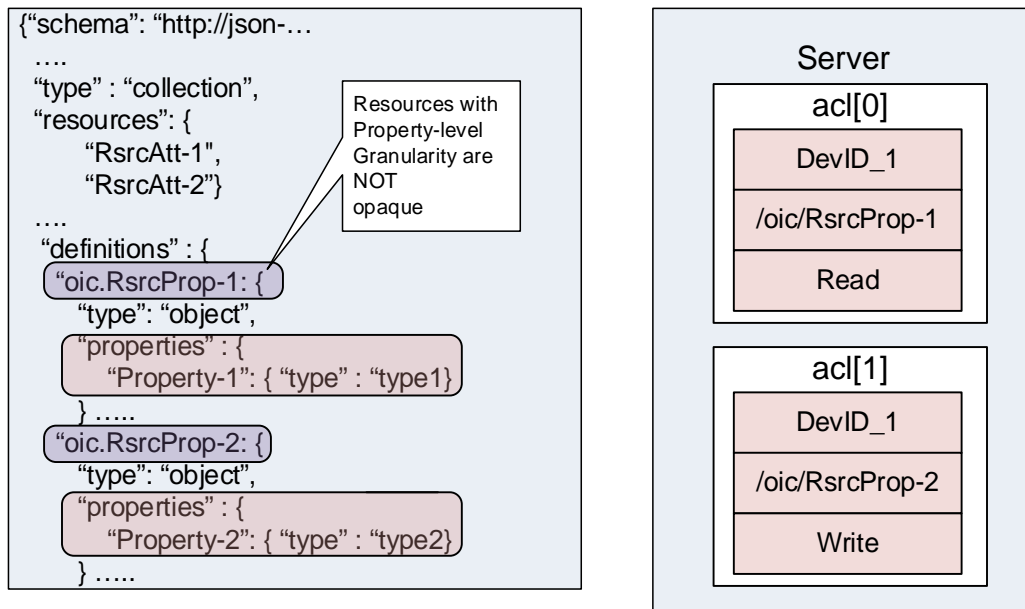


1167

1168

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

1169 Currently, OCF framework treats property level information as opaque; therefore, different  
 1170 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1  
 1171 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into  
 1172 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be  
 1173 achieved through use of Resource-level ACLs.



1174

1175

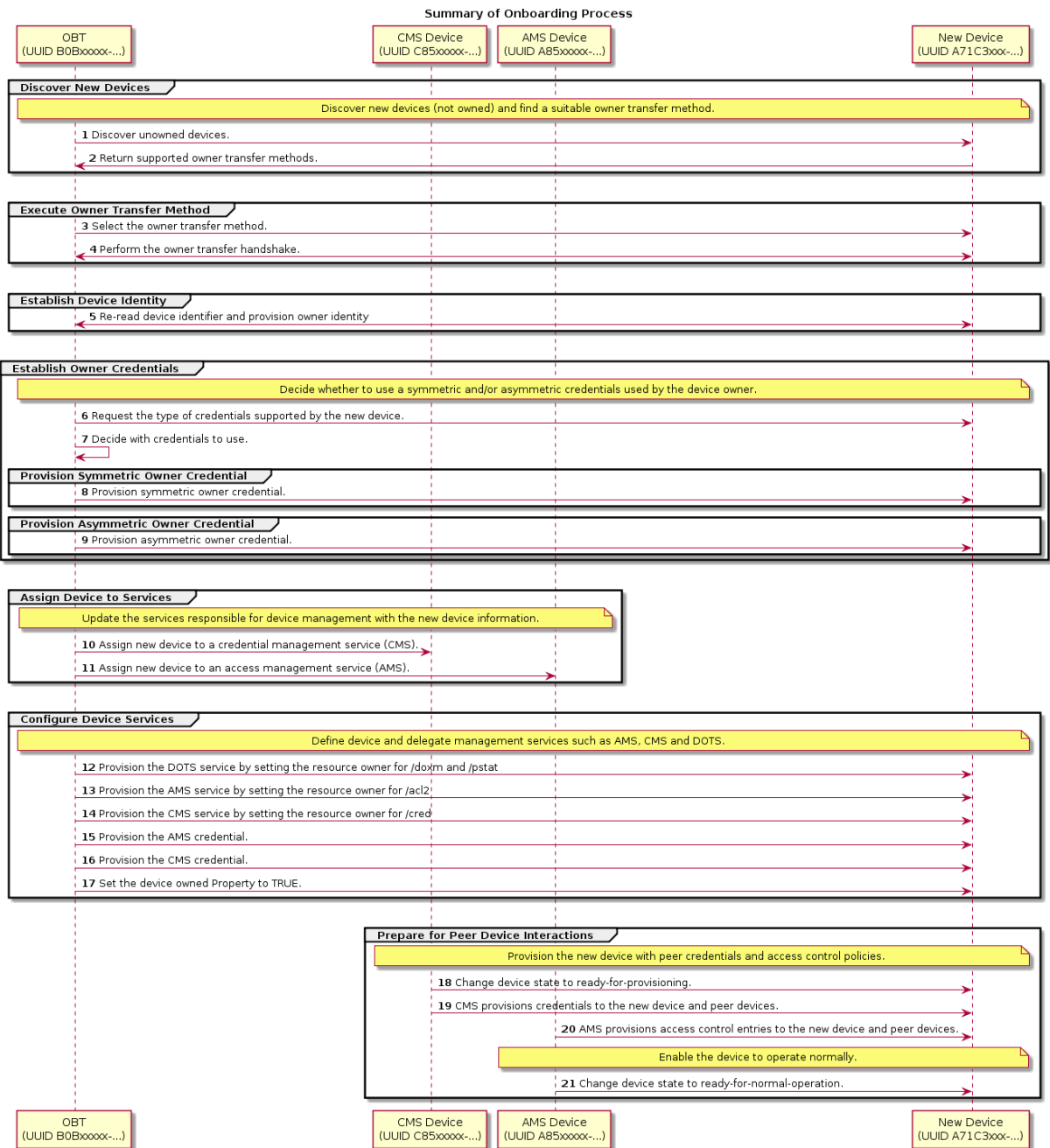
**Figure 9 – Property Level Access Control**

1176 **5.3 Onboarding Overview**

1177 **5.3.1 Onboarding General**

1178 Before a Device becomes operational in an OCF environment and is able to interact with other  
1179 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to  
1180 configure the ownership where the legitimate user that owns/purchases the Device uses an  
1181 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to  
1182 establish ownership. Once ownership is established, the OBT becomes the mechanism through  
1183 which the Device can then be provisioned, at the end of which the Device becomes operational  
1184 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an  
1185 OCF Device.

1186 Figure 10 depicts Onboarding Overview.



1187  
1188

**Figure 10 – Onboarding Overview**

1189 This clause explains the onboarding and security provisioning process but leaves the provisioning  
 1190 of non-security aspects to other OCF documents. In the context of security, all Devices are  
 1191 required to be provisioned with minimal security configuration that allows the Device to securely  
 1192 interact/communicate with other Devices in an OCF environment. This minimal security  
 1193 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified  
 1194 in 7.5.

1195 Onboarding and provisioning implementations could utilize services defined outside this  
 1196 document, it is expected that in using other services, trust between the device being onboarded  
 1197 and the various tools is not transitive. This implies that the device being onboarded will  
 1198 individually authenticate the credentials of each and every tool used during the onboarding

1199 process; that the tools not share credentials or imply a trust relationship where one has not been  
1200 established.

### 1201 **5.3.2 Onboarding Steps**

1202 The flowchart in Figure 11 shows the typical steps that are involved during onboarding. Although  
1203 onboarding may include a variety of non-security related steps, the diagram focus is mainly on  
1204 the security related configuration to allow a new Device to function within an OCF environment.  
1205 Onboarding typically begins with the Device becoming an Owned Device followed by configuring  
1206 the Device for the environment that it will operate in. This would include setting information such  
1207 as who can access the Device and what actions can be performed as well as what permissions  
1208 the Device has for interacting with other Devices.

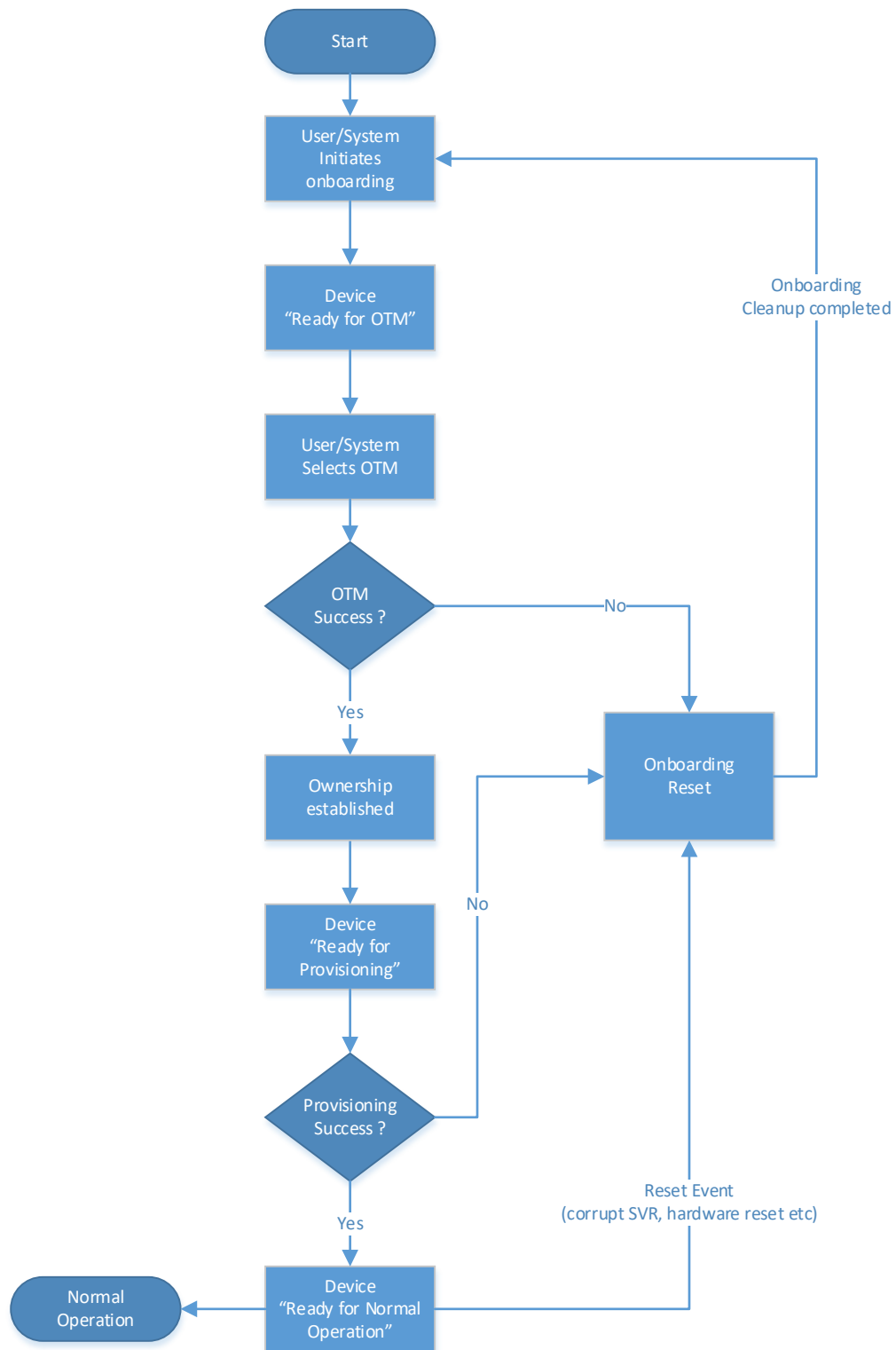


Figure 11 – OCF Onboarding Process

1209

1210

1211 **5.3.3 Establishing a Device Owner**

1212 The objective behind establishing Device ownership is to allow the legitimate user that  
 1213 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

1214 through the use of an OBT that includes the creation of an ownership context between the new  
1215 Device and the OBT tool and asserts operational control and management of the Device. The  
1216 OBT can be considered a logical entity hosted by tools/ Servers such as a network management  
1217 console, a device management tool, a network-authoring tool, a network provisioning tool, a  
1218 home gateway device, or a home automation controller. A physical device hosting the OBT will be  
1219 subject to some security hardening requirements, thus preserving integrity and confidentiality of  
1220 any credentials being stored. The tool/Server that establishes Device ownership is referred to as  
1221 the OBT.

1222 The OBT uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term  
1223 owner transfer is used since it is assumed that even for a new Device, the ownership is  
1224 transferred from the manufacturer/provider of the Device to the buyer/legitimate user of the new  
1225 Device.

1226 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.  
1227 Owner transfer establishes the following

- 1228 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"  
1229 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during  
1230 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property  
1231 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the  
1232 identity of Device as part of ownership transfer.
- 1233 – The Device owner establishes trust in the Device through the OTM.
- 1234 – Preparing the Device for provisioning by providing credentials that may be needed.

#### 1235 **5.3.4 Provisioning for Normal Operation**

1236 Once the Device has the necessary information to initiate provisioning, the next step is to  
1237 provision additional security configuration that allows the Device to become operational. This can  
1238 include setting various parameters and may also involve multiple steps. Also provisioning of  
1239 ACL's for the various Resources hosted by the Server on the Device is done at this time. The  
1240 provisioning step is not limited to this stage only. Device provisioning can happen at multiple  
1241 stages in the Device's operational lifecycle. However specific security related provisioning of  
1242 Resource and Property state would likely happen at this stage at the end of which, each Device  
1243 reaches the Onboarded Device "Ready for Normal Operation" State. The "Ready for Normal  
1244 Operation" State is expected to be consistent and well defined regardless of the specific OTM  
1245 used or regardless of the variability in what gets provisioned. However individual OTM  
1246 mechanisms and provisioning steps may specify additional configuration of Resources and  
1247 Property states. The minimal mandatory configuration required for a Device to be in "Ready for  
1248 Normal Operation" state is specified in 8.

#### 1249 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview**

1250 As mentioned in the start of clause 5, communication between a Device and OCF Cloud is  
1251 subject to different criteria in comparison to Devices which are within a single local network. The  
1252 Device is configured in order to connect to the OCF Cloud by a Mediator as specified in the  
1253 "CoAPCloudConf" Resource clauses in OCF Cloud Specification. Provisioning includes the  
1254 remote connectivity and local details such as URL where the OCF Cloud hosting environment can  
1255 be found and the OCF Cloud verifiable Access Token.

#### 1256 **5.3.6 OCF Compliance Management System**

1257 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that  
1258 provides Certification status and information for OCF Devices.

1259 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:  
1260 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1261 The OBT shall possess the Root Certificate needed to enable https connection to the URI  
1262 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1263 The OBT should periodically refresh its copy of the CPL via the URI  
1264 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security  
1265 Domain owner policy requirements.

## 1266 **5.4 Provisioning**

### 1267 **5.4.1 Provisioning General**

1268 In general, provisioning may include processes during manufacturing and distribution of the  
1269 Device as well as processes after the Device has been brought into its intended environment  
1270 (parts of onboarding process). In this document, security provisioning includes, processes after  
1271 ownership transfer (even though some activities during ownership transfer and onboarding may  
1272 lead to provisioning of some data in the Device) configuration of credentials for interacting with  
1273 provisioning services, configuration of any security related Resources and credentials for dealing  
1274 with any services that the Device need to contact later on.

1275 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to  
1276 be provisioned with proper security credentials and parameters for regular operation. These  
1277 parameters can include:

- 1278 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1279 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the  
1280 same OBT, but may be part of AMS in future.

1281 As mentioned, to accommodate a scalable and modular design, these functions are considered  
1282 as services that in future could be deployed as separate servers. Currently, the deployment  
1283 assumes that these services are all deployed as part of a OBT. Regardless of physical  
1284 deployment scenario, the same security-hardening requirement) applies to any physical server  
1285 that hosts the tools and security provisioning services discussed here.

1286 Devices are *aware* of their security provisioning status. Self-awareness allows them to be  
1287 proactive about provisioning or re-provisioning security Resources as needed to achieve the  
1288 devices operational goals.

### 1289 **5.4.2 Provisioning other services**

1290 To be able to support the use of potentially different device management service hosts, each  
1291 Device Secure Virtual Resource (SVR) has an associated Resource owner identified in the  
1292 Resource's rowneruuid Property.

1293 The DOTS shall update the rowneruuid Property of the "/oic/sec/doxm" and "/oic/sec/pstat"  
1294 resources with the DOTS resource owner identifier.

1295 The DOTS shall update the rowneruuid Property of the "/oic/sec/cred" resource with the CMS  
1296 resource owner identifier.

1297 The DOTS shall update the rowneruuid Property of the "/oic/sec/acl2" resource with the AMS  
1298 resource owner identifier

1299 When these OCF Services are configured, the Device may proactively request provisioning and  
1300 verify provisioning requests are authorized. The DOTS shall provision credentials that enable  
1301 secure connections between OCF Services and the new Device. The DOTS may initiate client-  
1302 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed  
1303 provisioning by setting tm Property of the "/oic/sec/pstat" Resource.



1304 **5.4.3 Provisioning Credentials for Normal Operation**

1305 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1306 – Pairwise symmetric keys
- 1307 – Group symmetric keys
- 1308 – Certificates
- 1309 – Raw asymmetric keys

1310 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS  
1311 credential provisioned by the DOTS.

1312 The following example describes how a Device updates a symmetric key credential involving a  
1313 peer Device. The Device discovers the credential to be updated; for example, a secure  
1314 connection attempt fails. The Device requests its CMS to supply the updated credential. The  
1315 CMS returns an updated symmetric key credential. The CMS updates the corresponding  
1316 symmetric key credential on the peer Device.

1317 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1318 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)  
1319 asserted by the Client requesting the Resource and compare that role identifier(s) with the  
1320 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned  
1321 with one or more role credentials.

1322 Each Device holds the role information as a Property within the credential Resource.

1323 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a  
1324 certificate role credential.

1325 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned  
1326 for a client, access to a Resource is granted if it would be granted under any of the roles.

1327 **5.4.5 ACL provisioning**

1328 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.  
1329 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL  
1330 policy by updating the Device's ACL Resources.

1331 The AMS shall digitally sign an ACL as part of issuing a "/oic/sec/sacl" Resource if the Device  
1332 supports the "/oic/sec/sacl" Resource. The public key used by the Device to verify the signature  
1333 shall be provisioned by the CMS as needed. A "/oic/sec/cred" Resource with an asymmetric key  
1334 type or signed asymmetric key type is used. The "PublicData" Property contains the AMS's public  
1335 key.

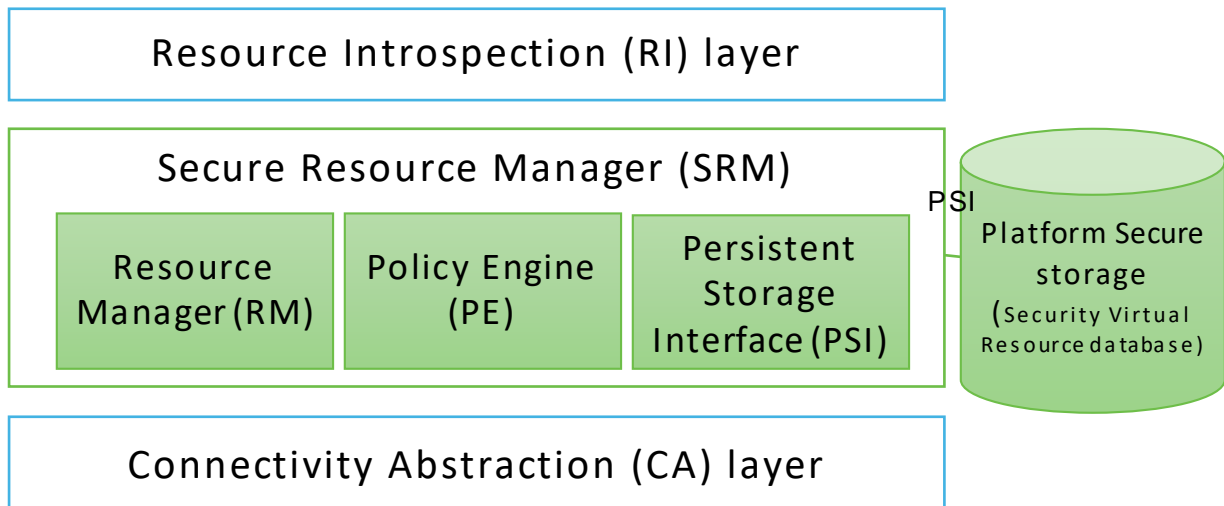
1336 **5.5 Secure Resource Manager (SRM)**

1337 SRM plays a key role in the overall security operation. In short, SRM performs both management  
1338 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3  
1339 main functional elements:

- 1340 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using  
1341 PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3)  
1342 Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a  
1343 format that is consistent with device-specific data store format. However, the RM will use  
1344 JSON format to marshal SVR data structures before be passed to PSI for storage, or travel  
1345 off-device.

- 1346 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control
- 1347 policies responds to the requests with either "ACCESS\_GRANTED" or "ACCESS\_DENIED".
- 1348 To make the access decisions, the PE consults the appropriate ACL and looks for best
- 1349 Access Control Entry (ACE) that can serve the request given the subject (Device or role) that
- 1350 was authenticated by DTLS.
- 1351 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in
- 1352 its own memory and storage. The SRM design is modular such that it may be implemented in
- 1353 the Platform's secure execution environment; if available.

1354 Figure 12 depicts OCF's SRM Architecture.



1355

1356 **Figure 12 – OCF's SRM Architecture**

1357 **5.6 Credential Overview**

1358 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional  
 1359 communication. Credentials can be symmetric or asymmetric. Each device stores secret and  
 1360 public parts of its own credentials where applicable, as well as credentials for other devices that  
 1361 have been provided by the DOTS or a CMS. These credentials are then used in the  
 1362 establishment of secure communication sessions (e.g. using DTLS) to validate the identities of  
 1363 the participating parties. Role credentials are used once an authenticated session is established,  
 1364 to assert one or more roles for a device.

1365 Access Tokens are provided to an OCF Cloud once an authenticated session with an OCF Cloud  
 1366 is established, to verify the User ID with which the Device is to be associated.

1367

## 1368 **6 Security for the Discovery Process**

### 1369 **6.1 Preamble**

1370 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,  
1371 called links) for the Resources hosted by the Server, complemented by attributes about those  
1372 Resources and possible further link relations. (in accordance to clause 10 in ISO/IEC 30118-  
1373 1:2018)

### 1374 **6.2 Security Considerations for Discovery**

1375 When defining discovery process, care must be taken that only a minimum set of Resources are  
1376 exposed to the discovering entity without violating security of sensitive information or privacy  
1377 requirements of the application at hand. This includes both data included in the Resources, as  
1378 well as the corresponding metadata.

1379 To achieve extensibility and scalability, this document does not provide a mandate on  
1380 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on  
1381 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any  
1382 of the Resources.

1383 The "/oic/sec/acl2" Resource contains ACL entries governing access to the Server hosted  
1384 Resources. (See 13.5)

1385 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery  
1386 process itself needs to be secured. This document sets the following requirements for the  
1387 discovery process:

- 1388 1) Providing integrity protection for discovered Resources.
- 1389 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1390 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)  
1391 on the known "/oic/res" Resource.

1392 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a  
1393 Server cannot determine the identity of the requester. In such cases, a Server that wants to  
1394 authenticate the Client before responding can list the secure discovery URI (e.g.  
1395 coaps://IP:PORT/oic/res ) in the unsecured "/oic/res" Resource response. This means the secure  
1396 discovery URI is by default discoverable by any Client. The Client will then be required to send a  
1397 separate unicast request using DTLS to the secure discovery URI.

1398 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to  
1399 "/oic/res" Resource if and only if the Client has permissions to perform at least one of the CRUDN  
1400 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1401 For example, a Client with Device Id "d1" makes a RETRIEVE request on the "/door" Resource  
1402 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1403 {  
1404   "aclist2": [  
1405     {  
1406       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1407       "resources": [{"href": "/door"}],  
1408       "permission": 2, // RETRIEVE  
1409       "aceid": 1  
1410     }  
1411   ],
```

```

1412     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1413 }
1414 {
1415     "aclist2": [
1416         {
1417             "subject": {"authority": "owner", "role": "owner"}
1418             "resources": [{"href": "/door"}],
1419             "permission": 2, // RETRIEVE
1420             "aceid": 2
1421         }
1422     ],
1423     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1424 }
1425 {
1426     "aclist2": [
1427         {
1428             "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1429             "resources": [{"href": "/door/lock"}],
1430             "permission": 4, // UPDATE
1431             "aceid": 3
1432         }
1433     ],
1434     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1435 }
1436 {
1437     "aclist2": [
1438         {
1439             "subject": {"conntype": "anon-clear"},
1440             "resources": [{"href": "/light"}],
1441             "permission": 2, // RETRIEVE
1442             "aceid": 4
1443         }
1444     ],
1445     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1446 }

```

1447 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when  
1448 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will  
1449 include the URI of the "/door" Resource metadata. Client "d2" will have access to both the  
1450 Resources. ACE2 will prevent "d4" from update.

1451 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1452 [
1453 {
1454     "href": "/door",
1455     "rt": ["oic.r.door"],
1456     "if": ["oic.if.b", "oic.if.ll"],

```

```
1457     "di": "0685B960-736F-46F7-BE C0-9E6 CBD61ADC1",
1458   }
1459 ]
```

1460 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1461 [
1462   {
1463     "href": "/door",
1464     "rt": ["oic.r.door"],
1465     "if": ["oic.if.b", "oic.if.ll"],
1466     "di": "0685B960-736F-46F7-BE C0-9E6 CBD61ADC1"
1467   },
1468   {
1469     "href": "/door/lock",
1470     "rt": ["oic.r.lock"],
1471     "if": ["oic.if.b"],
1472     "type": ["application/json", "application/exi+xml"]
1473   }
1474 ]
```

1475 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

```
1476 [
1477   {
1478     "href": "/door/lock",
1479     "rt": ["oic.r.lock"],
1480     "if": ["oic.if.b"],
1481     "type": ["application/json", "application/exi+xml"],
1482     "di": "0685B960-736F-46F7-BE C0-9E6 CBD61ADC1"
1483   }
1484 ]
```

1485 Discovery results delivered to any device regarding d3's "/oic/res" Resource from the unsecure  
1486 interface:

```
1487 [
1488   {
1489     "di": "0685B960-736F-46F7-BE C0-9E6 CBD61ADC1",
1490     "href": "/light",
1491     "rt": ["oic.r.light"],
1492     "if": ["oic.if.s"]
1493   }
1494 ]
```

1495

1496 **7 Security Provisioning**

1497 **7.1 Device Identity**

1498 **7.1.1 General Device Identity**

1499 Each Device, which is a logical device, is identified with a Device ID.

1500 Devices shall be identified by a Device ID value that is established as part of device onboarding.  
1501 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. "urn:uuid"). Device IDs shall  
1502 be unique within the scope of operation of the corresponding OCF Security Domain, and should  
1503 be universally unique. The DOTS shall ensure Device ID of the new Device is unique within the  
1504 scope of the owner's OCF Security Domain. The DOTS shall verify the chosen new device  
1505 identifier does not conflict with Device IDs previously introduced into the OCF Security Domain.

1506 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"  
1507 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying  
1508 authentication credentials of a peer device.

1509 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of  
1510 credentials, both its own and other Device credentials, in the "/oic/sec/cred" Resource. The  
1511 device ID can be used to distinguish between a device's own credential, and credentials for other  
1512 devices. Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the  
1513 device.

1514 Device ID shall be:

- 1515 – Unique
- 1516 – Immutable
- 1517 – Verifiable

1518 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the  
1519 device as described later in this clause.

1520 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The  
1521 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in  
1522 the device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1523 An OCF Platform may have a secure execution environment, which shall be used to secure  
1524 unique identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to  
1525 provide each Device with the appropriate and separate security.

1526 **7.1.2 Device Identity for Devices with UAID [Deprecated]**

1527 This clause is intentionally left blank.

1528 **7.2 Device Ownership**

1529 This is an informative clause. Devices are logical entities that are security endpoints that have an  
1530 identity that is authenticable using cryptographic credentials. A Device is Unowned when it is first  
1531 initialized. Establishing device ownership is a process by which the device asserts its identity to  
1532 the DOTS and the DOTS provisions an owner identity. This exchange results in the device  
1533 changing its ownership state, thereby preventing a different DOTS from asserting administrative  
1534 control over the device.

1535 The ownership transfer process starts with the OBT discovering a new device that is in Unowned  
1536 state through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new  
1537 device. At the end of ownership transfer, the following is accomplished:

- 1538 1) The DOTS shall establish a secure session with new device.
- 1539 2) Optionally asserts any of the following:
- 1540 a) Proximity (using PIN) of the OBT to the Platform.
- 1541 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific
- 1542 attributes.
- 1543 3) Determines the device identifier.
- 1544 4) Determines the device owner.
- 1545 5) Specifies the device owner (e.g. Device ID of the OBT).
- 1546 6) Provisions the device with owner's credentials.
- 1547 7) Sets the "Owned" state of the new device to TRUE.

1548 NOTE A Device which connects to the OCF Cloud still retains the ownership established at onboarding with the

1549 DOTS.

### 1550 **7.3 Device Ownership Transfer Methods**

#### 1551 **7.3.1 OTM implementation requirements**

1552 This document provides specifications for several methods for ownership transfer.

1553 Implementation of each individual ownership transfer method is considered optional. However,

1554 each device shall implement at least one of the ownership transfer methods not including vendor

1555 specific methods.

1556 All OTMs included in this document are considered optional. Each vendor is required to choose

1557 and implement at least one of the OTMs specified in this document. The OCF, does however,

1558 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability

1559 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with

1560 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the

1561 preferred approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in

1562 designing vendor-specific OTMs.

1563 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer

1564 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.

1565 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the

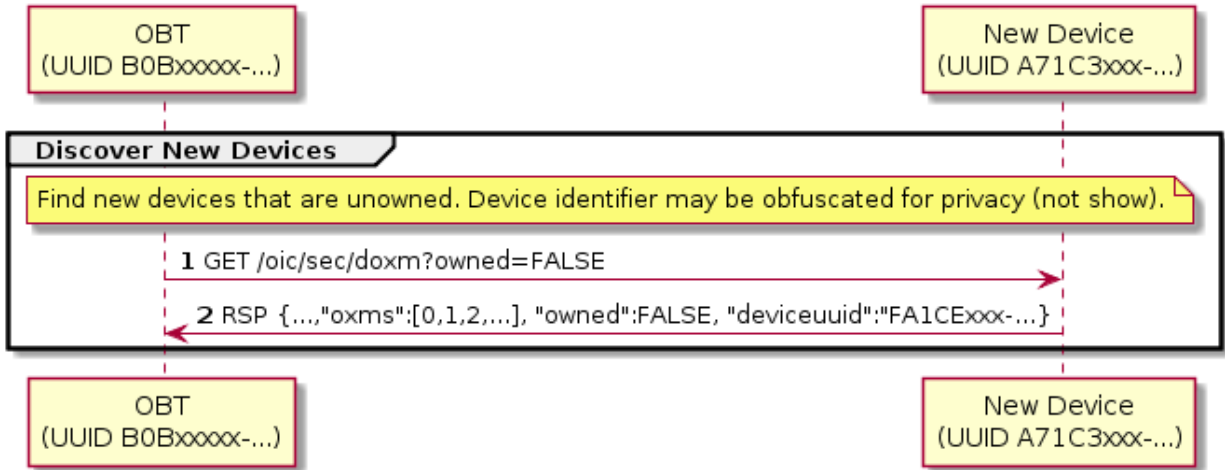
1566 "/oic/sec/doxm" Resource. The DOTS shall query the Device's supported credential types using

1567 the credtypes Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision

1568 credentials according to the credential types supported.

1569 Figure 13 depicts new Device discovery sequence.

### Discover New Devices Sequence



1570

1571

**Figure 13 – Discover New Device Sequence**

1572

1573

**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.

1574 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs  
 1575 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for  
 1576 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the  
 1577 new Device.

1578 The new device may have to perform some initialization steps at the beginning of an OTM. For  
 1579 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN  
 1580 value. The OBT shall POST to the oxmsel property of "/oic/sec/doxm" the value corresponding to  
 1581 the OTM being used, before performing other OTM steps. This POST notifies the new device that  
 1582 ownership transfer is starting.

1583 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT  
 1584 and the OBT to authenticate to the new device.

1585 The DOTS may perform additional provisioning steps subsequent to owner transfer success  
 1586 leveraging the established OTM session.

1587 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall  
 1588 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array  
 1589 includes a SVR.



1590 **7.3.2 SharedKey Credential Calculation**

1591 The SharedKey credential is derived using a PRF that accepts the key\_block value resulting from  
1592 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following  
1593 calculation to ensure interoperability across vendor products:

1594 SharedKey = *PRF*(Secret, Message);

1595 Where:

- 1596 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1597 - Secret is the key\_block resulting from the DTLS handshake
  - 1598 ▪ See IETF RFC 5246 clause 6.3
  - 1599 ▪ The length of key\_block depends on cipher suite.
    - 1600 • (e.g. 96 bytes for TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
    - 1601 40 bytes for TLS\_PSK\_WITH\_AES\_128\_CCM\_8)
- 1602 - Message is a concatenation of the following:
  - 1603 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
    - 1604 • See clause 13.2.4 for specific DoxmTypes
  - 1605 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
    - 1606 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
  - 1607 ▪ Device ID is new device's UUID Device ID
    - 1608 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1609 - SharedKey Length will be 32 octets.
  - 1610 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the left most 16 octets will be used.
  - 1611 DTLS sessions using 256-bit encryption cipher suites will use all 32 octets.

1612 **7.3.3 Certificate Credential Generation**

1613 The Certificate Credential will be used by Devices for secure bidirectional communication. The  
1614 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used  
1615 to mutually establish the authenticity of the Device. The onboarding details for certificate  
1616 generation will be specified in a later version of this document.

1617 **7.3.4 Just-Works OTM**

1618 **7.3.4.1 Just-Works OTM General**

1619 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a  
1620 secure connection through which a device should be provisioned for use within the owner's OCF  
1621 Security Domain. Provisioning additional credentials and Resources is a typical step following  
1622 ownership establishment. The pre-shared key is called SharedKey.

1623 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite  
1624 defined for the Just-works OTM.

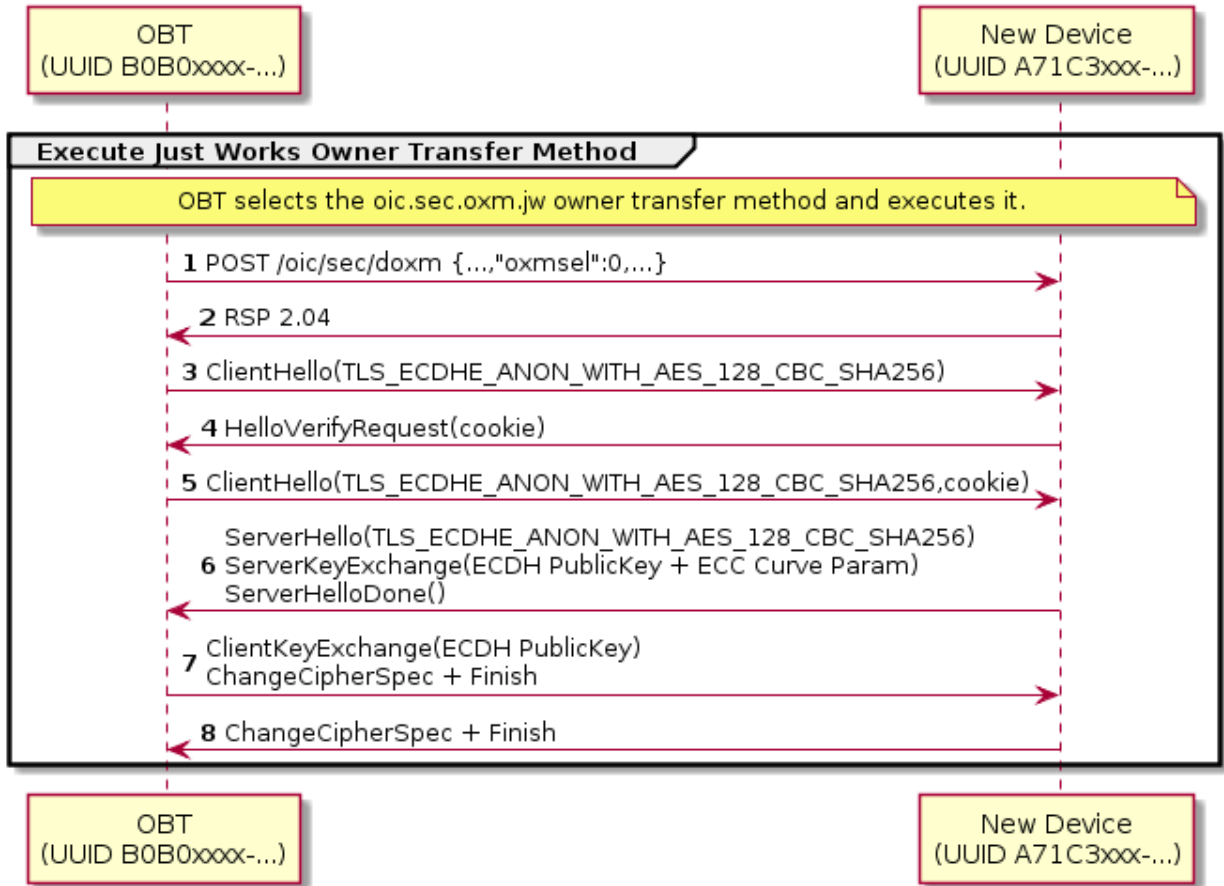
1625 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1626 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,  
1627 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

1628 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for  
1629 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1630 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.

### Perform Just-Works Owner Transfer Method



1631

1632

**Figure 14 – A Just Works OTM**

1633

**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 an 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.

#### 1634 **7.3.4.2 Security Considerations**

1635 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this  
 1636 method presumes that both the OBT and the new device perform the "just-works" method  
 1637 assumes onboarding happens in a relatively safe environment absent of an attack device.

1638 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to  
 1639 the device.

1640 The new device should use a temporal device ID prior to transitioning to an owned device while it  
 1641 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-  
 1642 temporal device ID that could differ from the temporal value during the secure session in which

1643 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not  
 1644 conflict with a Device ID already in use. If it is already in use the existing credentials are used to  
 1645 establish a secure session.

1646 An un-owned Device that also has established device credentials might be an indication of a  
 1647 corrupted or compromised device.

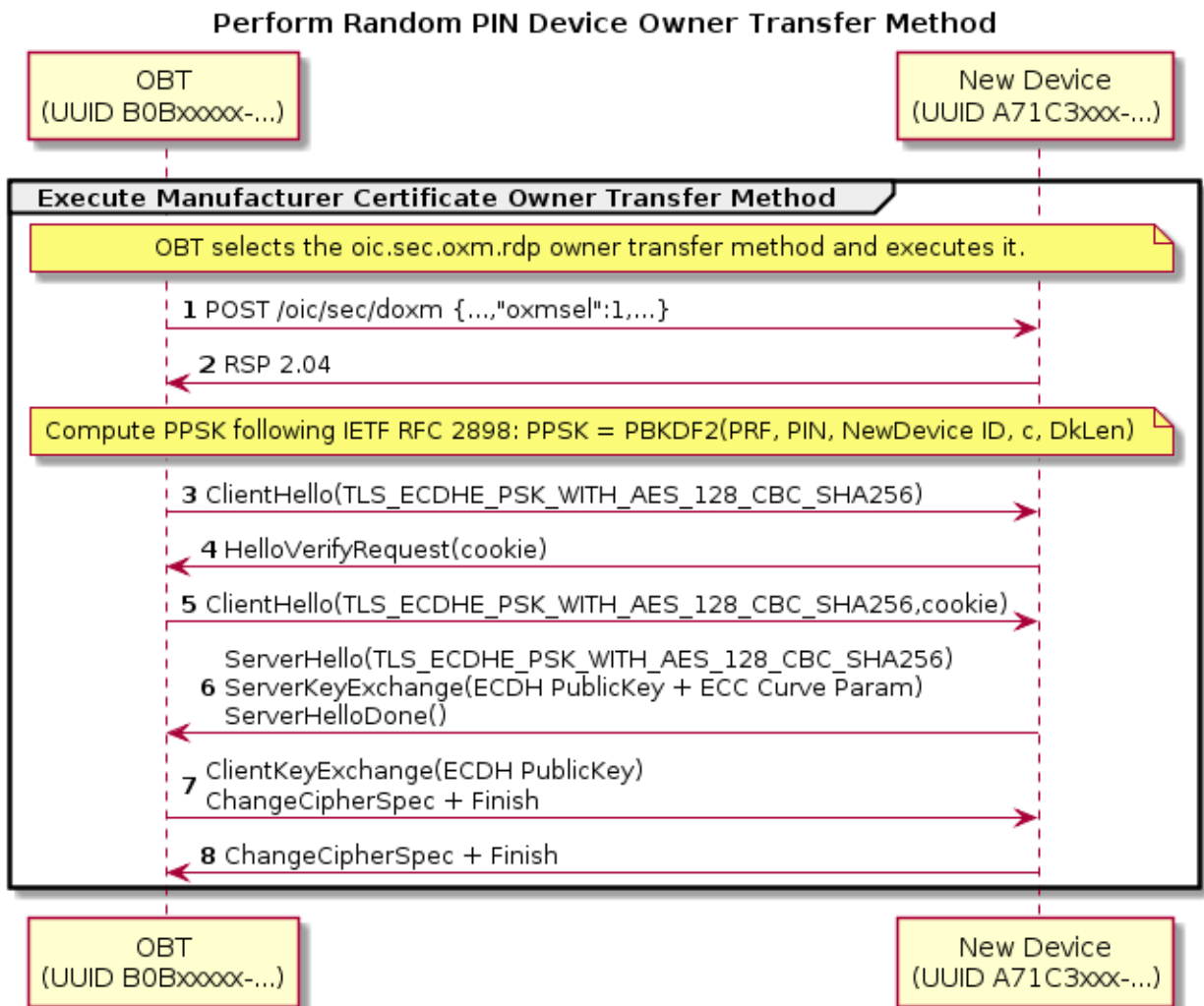
1648 **7.3.5 Random PIN Based OTM**

1649 **7.3.5.1 Random PIN OTM General**

1650 The Random PIN method establishes physical proximity between the new device and the OBT  
 1651 can prevent man-in-the-middle attacks. The Device generates a random number that is  
 1652 communicated to the OBT over an out-of-band channel. The definition of out-of-band  
 1653 communications channel is outside the scope of the definition of device OTMs. The OBT and new  
 1654 Device use the PIN in a key exchange as evidence that someone authorized the transfer of  
 1655 ownership by having physical access to the new Device via the out-of-band-channel.

1656 **7.3.5.2 Random PIN Owner Transfer Sequence**

1657 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.



1658

1659

**Figure 15 – Random PIN-based OTM**

1660

1661

**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.

1662 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF  
 1663 RFC 2898 and a PIN exchanged via an out-of-band method to generate a pre-shared key. The  
 1664 PIN-authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a PSK.

1665 
$$PPSK = PBKDF2(PR, PIN, Device\ ID, c, dkLen)$$

1666 The PBKDF2 function has the following parameters:

- 1667 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.
- 1668 - PIN – obtain via out-of-band channel.
- 1669 - Device ID – UUID of the new device.

1670 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2

- 1671 - c – Iteration count initialized to 1000
- 1672 - dkLen – Desired length of the derived PSK in octets.

1673 **7.3.5.3 Security Considerations**

1674 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with  
 1675 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials  
 1676 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,  
 1677 allows an attacker to masquerade as the onboarded device.

1678 It is recommended that the entropy of the PIN be enough to withstand an online brute-force  
 1679 attack, 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-  
 1680 9a-z), or a 7-character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack  
 1681 (MITM) is when the attacker is active on the network and can intercept and modify messages  
 1682 between the OBT and device. In the MITM attack, the attacker must recover the PIN from the key  
 1683 exchange messages in "real time", i.e., before the peer's time out and abort the connection  
 1684 attempt. Having recovered the PIN, he can complete the authentication step of key exchange.  
 1685 The guidance given here calls for a minimum of 40 bits of entropy, however, the assurance this  
 1686 provides depends on the resources available to the attacker. Given the parallelizable nature of a  
 1687 brute force guessing attack, the attack enjoys a linear speedup as more cores/threads are added.  
 1688 A more conservative amount of entropy would be 64 bits. Since the Random PIN OTM requires  
 1689 using a DTLS ciphersuite that includes an ECDHE key exchange, the security of the Random PIN  
 1690 OTM is always at least equivalent to the security of the JustWorks OTM.

1691 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN.  
 1692 The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess  
 1693 in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an  
 1694 effective way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify  
 1695 the reduction, since an X-fold increase in time spent by the honest peers does not directly  
 1696 translate to an X-fold increase in time by the attacker. This asymmetry is because the attacker  
 1697 may use specialized implementations and hardware not available to honest peers. For this

1698 reason, when deciding how much entropy to use for a PIN, it is recommended that implementers  
1699 assume PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

1700 The Random PIN device OTM security depends on an assumption that a secure out-of-band  
1701 method for communicating a randomly generated PIN from the new device to the OBT exists. If  
1702 the OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN,  
1703 and the attacks described above apply. The out-of-band mechanism should be chosen such that  
1704 it requires proximity between the OBT and the new device. The attacker is assumed to not have  
1705 compromised the out-of-band-channel. As an example OOB channel, the device may display a  
1706 PIN to be entered into the OBT software. Another example is for the device to encode the PIN as  
1707 a 2D barcode and display it for a camera on the OBT device to capture and decode.

### 1708 **7.3.6 Manufacturer Certificate Based OTM**

#### 1709 **7.3.6.1 Manufacturer Certificate Based OTM General**

1710 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the  
1711 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device  
1712 and the OBT.

1713 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust  
1714 anchor.

1715 For some environments, policies or administrators, additional information about device  
1716 characteristics may be sought. This list of additional attestations that OCF may or may not have  
1717 tested (understanding that some attestations are incapable of testing or for which testing may be  
1718 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3  
1719 extension described in 9.4.2.2.6.

1720 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with  
1721 certificate data to authenticate their identities with the OBT in the process of bringing a new  
1722 device into operation on an OCF Security Domain. The onboarding process involves several  
1723 discrete steps:

#### 1724 1) Pre-on-board conditions

1725 a) The credential element of the Device's credential Resource ("/oic/sec/cred") containing the  
1726 manufacturer certificate shall be identified by the properties:

1727 i) the subject Property shall refer to the Device

1728 ii) the credusage Property shall contain the string "oic.sec.cred.mfgcert" to indicate that  
1729 the credential contains a manufacturer certificate

1730 b) The manufacturer certificate chain shall be contained in the identified credential element's  
1731 publicdata Property.

1732 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1733 d) If the device requires authentication of the OBT as part of ownership transfer, it is  
1734 presumed that the OBT has been registered and has obtained a certificate for its unique  
1735 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.

1736 e) User has configured the OBT app with network access info and account info (if any).

1737 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the  
1738 Device may authenticate the OBT to verify the OBT signature.

1739 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready  
1740 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted  
1741 link in accordance with the negotiated cipher suite.

1742 **7.3.6.2 Certificate Profiles**

1743 See 9.4.2 for details.

1744 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

1745 In order for full, mutual authentication to occur between the device and the OBT, both the device  
1746 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This  
1747 implies that OCF may need to obtain services from a Certificate Authority (e.g. Symantec,  
1748 Verisign, etc.) to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors  
1749 are derived.

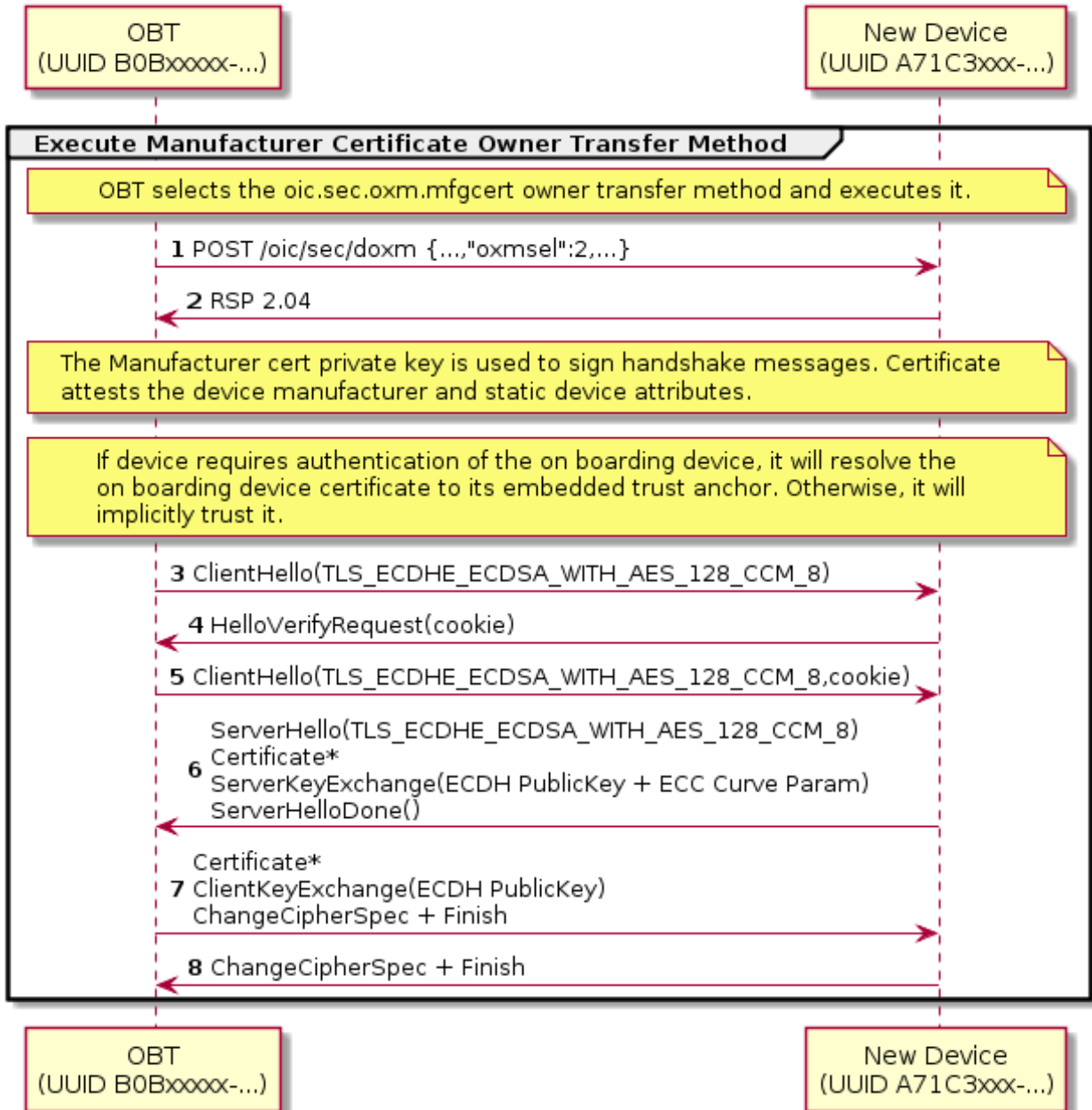
1750 The OBT shall authenticate the device during onboarding. However, the device is not required to  
1751 authenticate the OBT due to potential resource constraints on the device.

1752 In the case where the Device does NOT authenticate the OBT software, there is the possibility of  
1753 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,  
1754 which can compromise OCF Security Domain access credentials and/or personal information.

1755 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1756 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.

## Perform Manufacturer Certificate Owner Transfer Method



1757

1758

1759

1760

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

**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.

1761 **7.3.6.5 Security Considerations**

1762 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of  
1763 assurance that the private key cannot be compromised.

1764 The Platform manufacturer issues the manufacturer certificate and attests the private key  
1765 protection mechanism.

1766 **7.3.7 Vendor Specific OTMs**

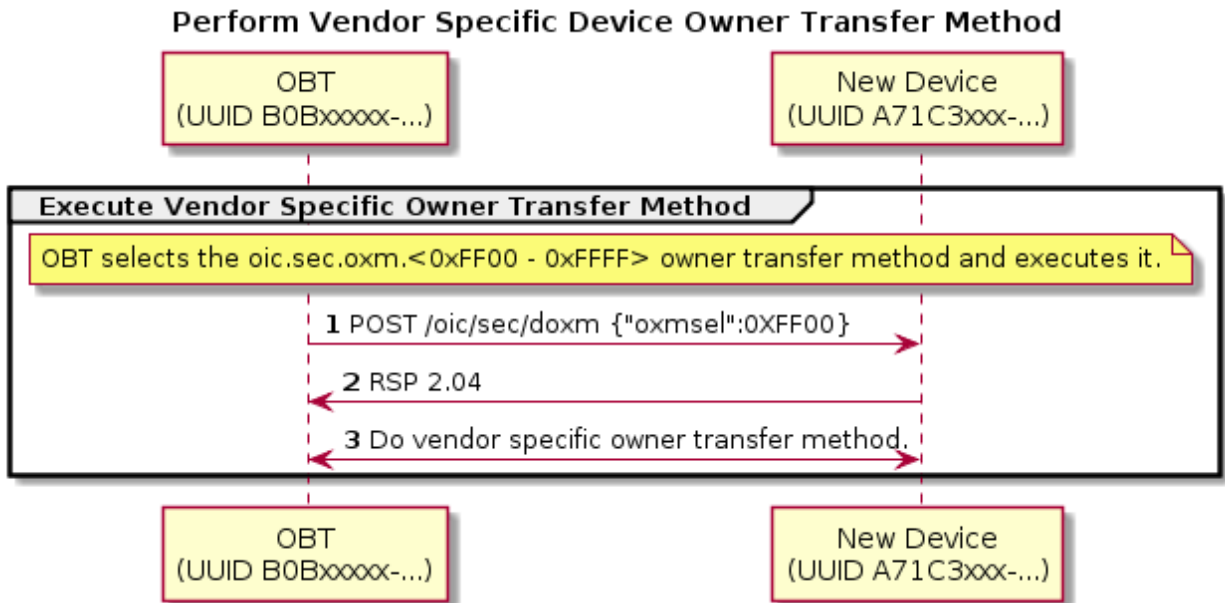
1767 **7.3.7.1 Vendor Specific OTM General**

1768 The OCF anticipates situations where a vendor will need to implement an OTM that  
1769 accommodates manufacturing or Device constraints. The Device OTM resource is extensible for  
1770 this purpose. Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1771 – The OBT must determine which credential types are supported by the Device. This is  
1772 accomplished by querying the Device's "/oic/sec/doxm" Resource to identify supported  
1773 credential types.
- 1774 – The OBT provisions the Device with OC(s).
- 1775 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1776 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security  
1777 Domain subsequent to ownership establishment.
- 1778 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks  
1779 to be performed by a security service.

1780 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1781 Vendor-specific OTM sequence example is shown in Figure 17 and steps described in Table 5.



1782

1783

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

1784



1785

**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

1786 **7.3.7.3 Security Considerations**

1787 The vendor is responsible for considering security threats and mitigation strategies.

1788 **7.3.8 Establishing Owner Credentials**

1789 Once the OBT and the new Device have authenticated and established an encrypted connection  
1790 using one of the defined OTM methods.

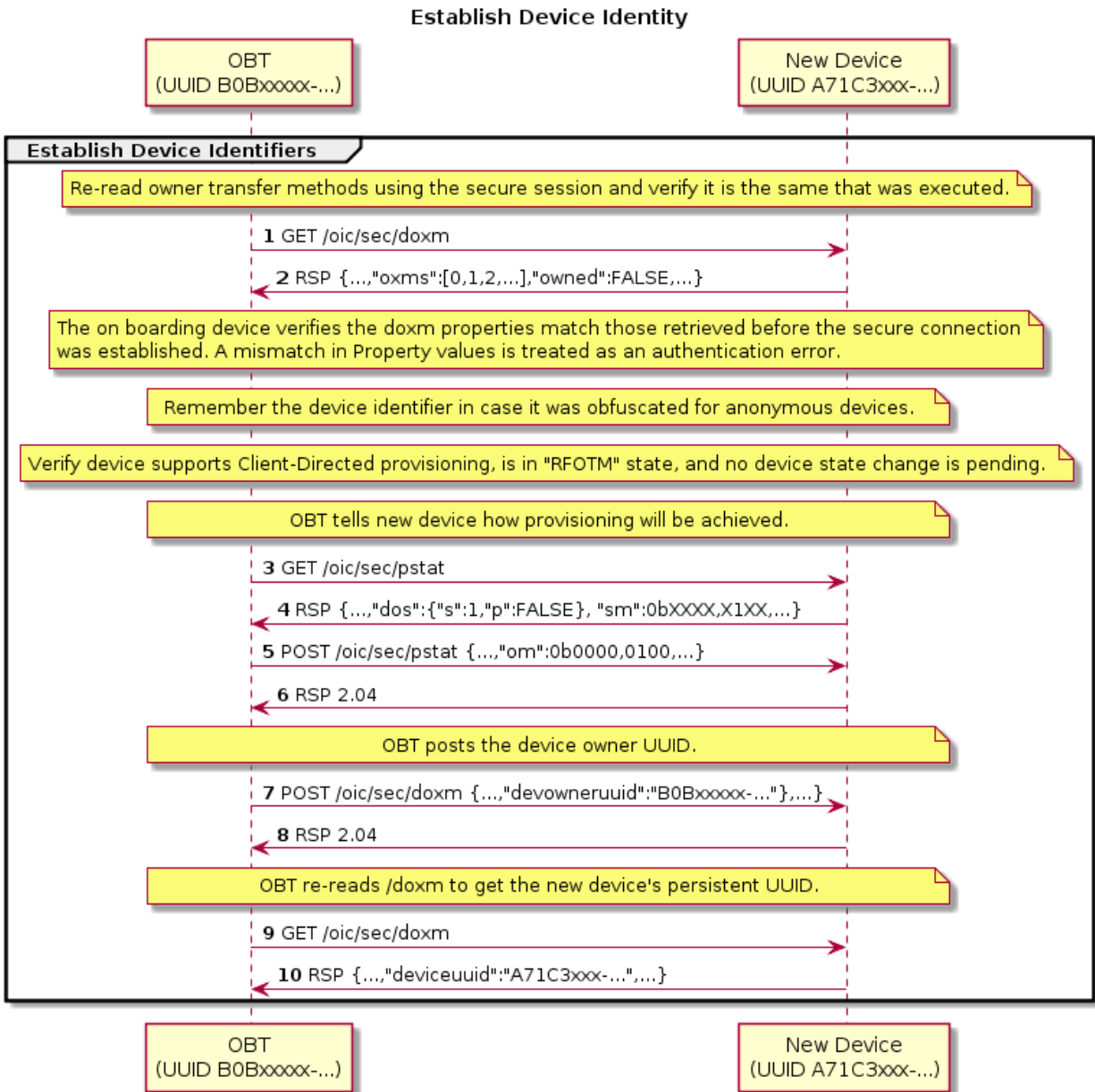
1791 Owner credentials may consist of certificates signed by the OBT or other authority, OCF Security  
1792 Domain access information, provisioning functions, shared keys, or Kerberos tickets.

1793 The OBT might then provision the new Device with additional credentials for Device management  
1794 and Device-to-Device communications. These credentials may consist of certificates with  
1795 signatures, UAID based on the Device public key, PSK, etc.

1796 The steps for establishing Device's owner credentials (OC) are:

- 1797 1) The OBT shall establish the Device ID and Device owner uuid - Figure 18 and Table 6  
1798 2) The OBT then establishes Device's OC - Figure 19 and Table 7. This can be either:  
1799 a) Symmetric credential - Figure 20 and Table 8.  
1800 b) Asymmetric credential - Figure 21 and Table 9.  
1801 3) Configure Device services - Figure 22 and Table 10.  
1802 4) Configure Device for peer to peer interaction - Figure 23 and Table 11.

1803



1804

1805

1806

1807

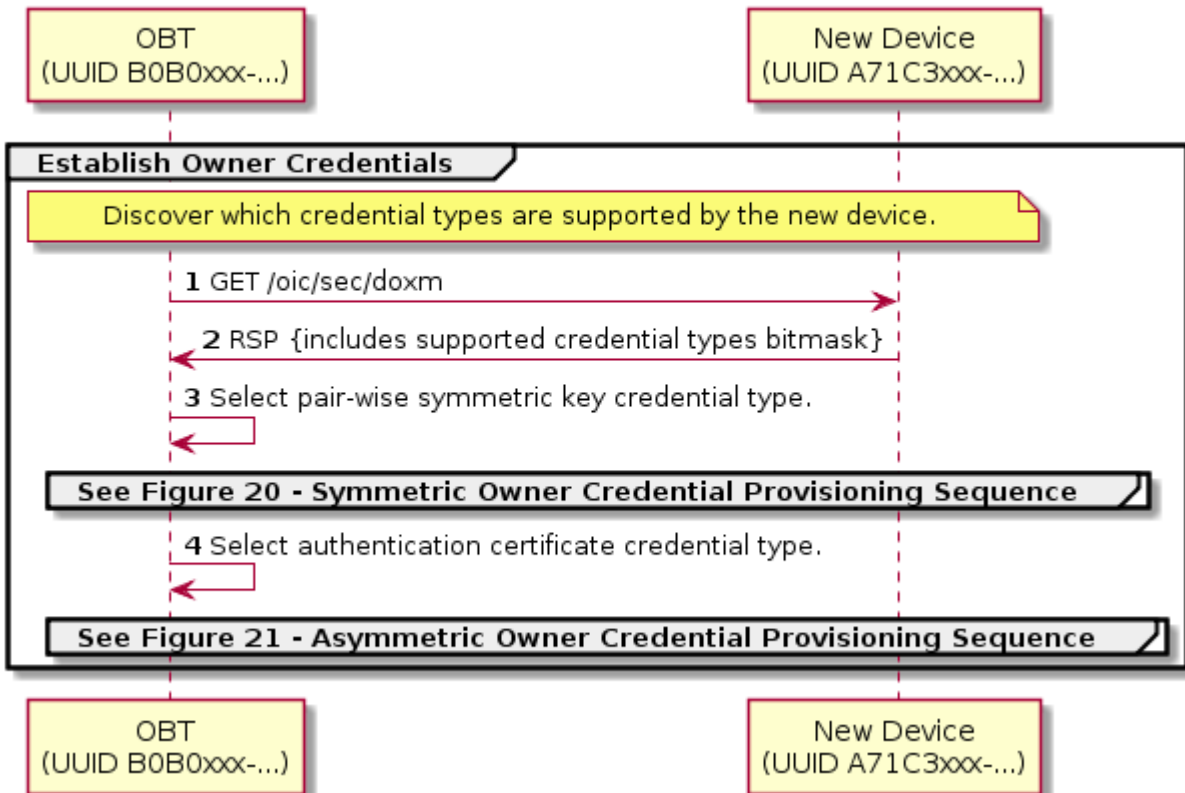
**Figure 18 – Establish Device Identity Flow**

**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



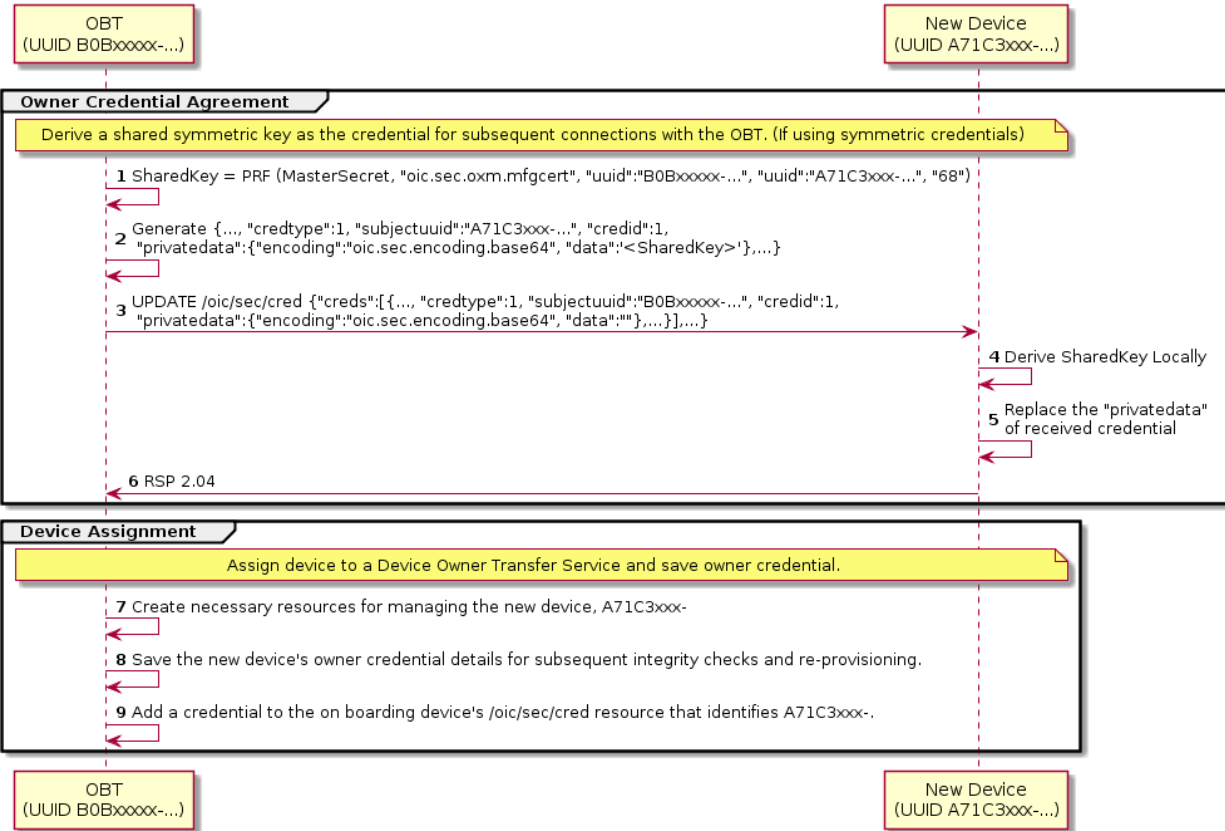
1808  
1809  
1810  
1811

**Figure 19 – Owner Credential Selection Provisioning Sequence**

**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.

### Symmetric Owner Credential (OC) Assignment Sequence



1812

1813

1814

1815

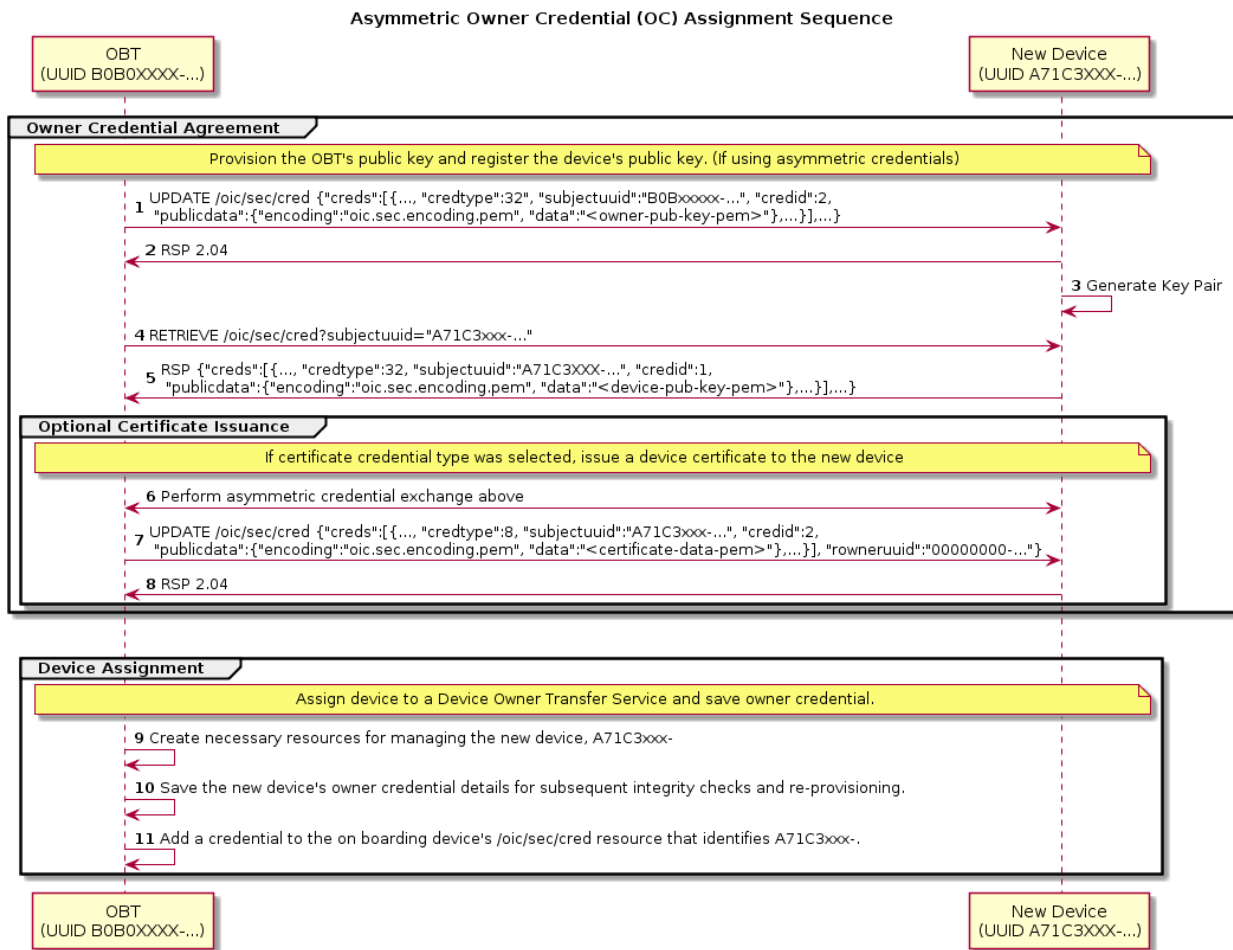
**Figure 20 – Symmetric Owner Credential Provisioning Sequence**

**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 its own "/oic/sec/cred" resource with the owner credential for

new device. Credential type is SYMMETRIC KEY.

- 1816 In particular, if the OBT selects symmetric owner credentials:
- 1817 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
  - 1818 described in 7.3.2.
  - 1819 – The OBT shall send an empty key to the new Device's "/oic/sec/cred" Resource, identified as
  - 1820 a symmetric pair-wise key.
  - 1821 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
  - 1822 generate the Shared Key using the SharedKey Credential Calculation method described in
  - 1823 7.3.2 and store it with the owner credential.
  - 1824 – The new Device shall use the Shared Key owner credential(s) stored via the "/oic/sec/cred"
  - 1825 Resource to authenticate the owner during subsequent connections.



1826  
1827 **Figure 21 – Asymmetric Owner Credential Provisioning Sequence**

1828  
1829 **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

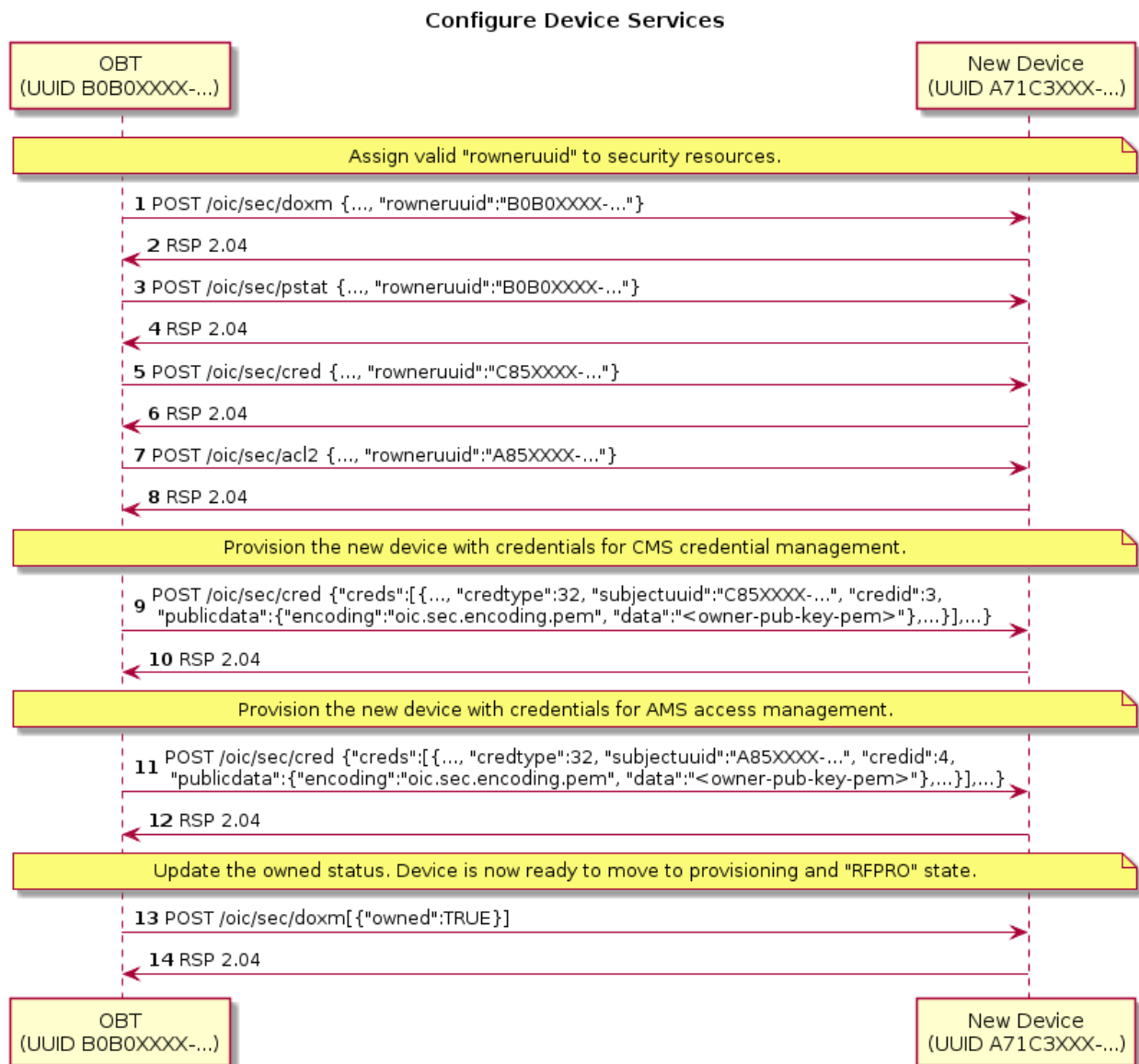
	Resource Property set with its public key (OC) to the new Device. It may be used subsequently to authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric 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 its own "/oic/sec/cred resource" with the owner credential for new device. Credential type is PUBLIC KEY.
12	(optional) The onboarding service provisions its own "/oic/sec/cred" resource with the owner credential for new device. Credential type is CERTIFICATE.

1830 If the OBT selects asymmetric owner credentials:

- 1831 – The OBT shall add its public key to the new Device's "/oic/sec/cred" Resource, identified as  
1832 an Asymmetric Encryption Key.
- 1833 – The OBT shall query the "/oic/sec/cred" Resource from the new Device, supplying the new  
1834 Device's UUID via the SubjectID query parameter. In response, the new Device shall return  
1835 the public Asymmetric Encryption Key, which the OBT shall retain for future owner  
1836 authentication of the new Device.

1837 If the OBT selects certificate owner credentials:

- 1838 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the  
1839 public key returned by the new Device, signed by a mutually-trusted CA, and complying with  
1840 the Certificate Credential Generation requirements defined in 7.3.3.
- 1841 – The OBT shall add the newly-created certificate chain to the "/oic/sec/cred" Resource,  
1842 identified as an Asymmetric Signing Key with Certificate.



1843

1844

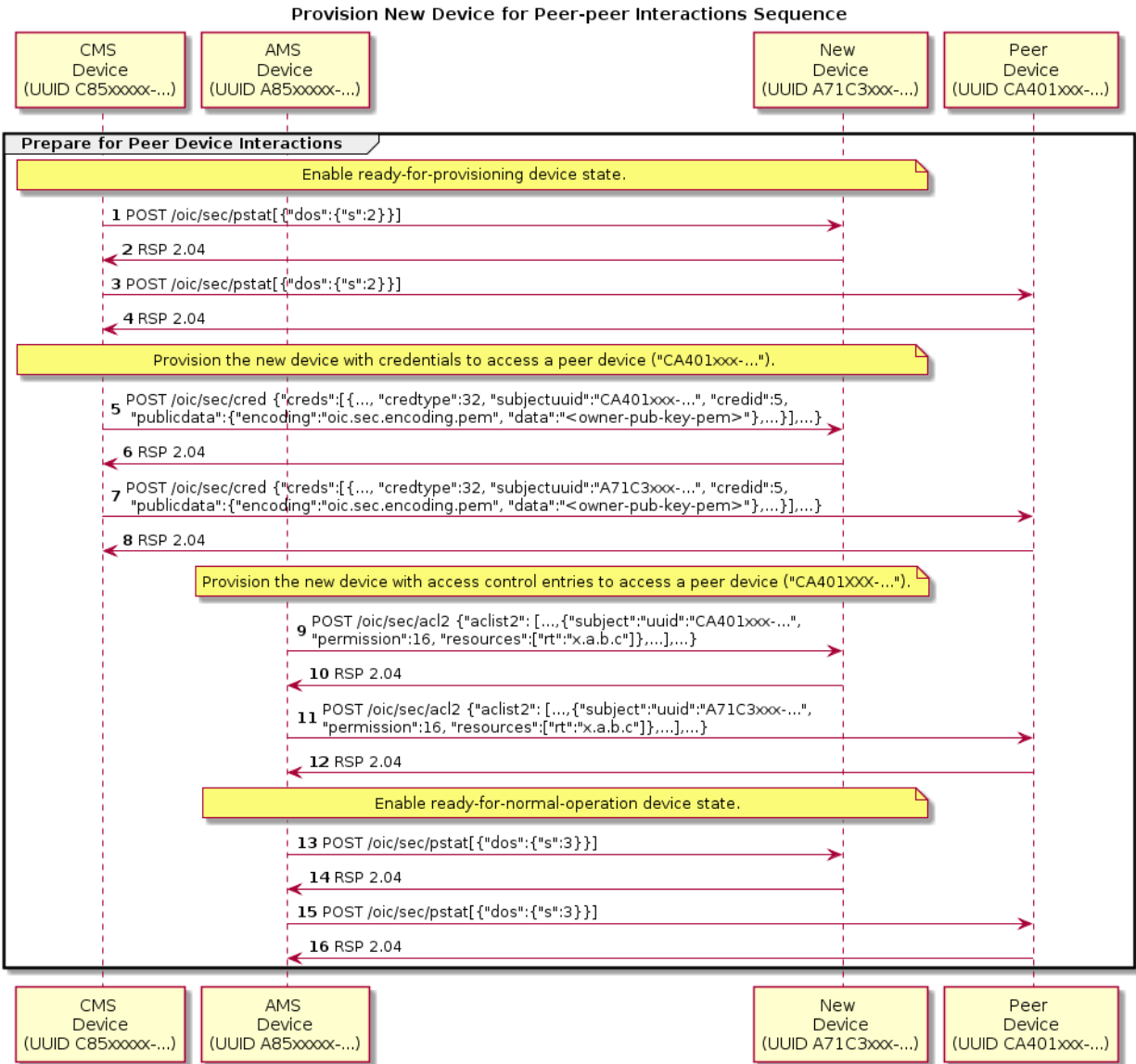
1845

1846

**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.



1847

1848

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

1849

**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.



1850 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1851 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are  
1852 acceptable when transferring ownership of a new Device. Some of the factors to be considered  
1853 when determining those requirements are:

- 1854 – The security considerations described for each of the OTMs
- 1855 – The probability that a man-in-the-middle attacker might be present in the environment used to  
1856 perform the ownership transfer

1857 For example, the operator of an OBT might require that all of the Devices being onboarded  
1858 support either the Random PIN or the Manufacturer Certificate OTM.

1859 When such a local OTM policy exists, the OBT should try to use just the OTMs that are  
1860 acceptable according to that policy, regardless of the doxm contents obtained during step 1 from  
1861 the sequence diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an  
1862 unauthenticated and/or unencrypted connection between the OBT and the Device, the contents of  
1863 the response to the GET request might have been tampered by a man-in-the-middle attacker. For  
1864 example, the list of OTMs supported by the new Device might have been altered by the attacker.

1865 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new  
1866 Device to fail. In such cases, the OBT has no way of determining if the session failed because  
1867 the new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle  
1868 injected such a failure into the communication between the OBT and the new Device.

1869 The current version of this document leaves the design and user experience related to the OTM  
1870 policy as OBT implementation details.

1871 **7.3.10 Security Profile Assignment**

1872 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results  
1873 could be accessed from a manufacturer's certificate, OCF web server or other public repository.  
1874 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device  
1875 is authorized to possess and configures the Device with the subset of evaluated security profiles  
1876 best suited for the OCF Security Domain owner's intended segmentation strategy.

1877 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles"  
1878 Property of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification  
1879 process. The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the  
1880 values contained in the "supportedprofiles". The manufacturer default value shall be re-asserted  
1881 when the Device transitions to RESET Device State.

1882 The OCF Device shall only allow the "/oic/sec/sp" Resource to be updated when the Device is in  
1883 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as  
1884 directed by a Security Profile.

1885 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a  
1886 subset of the OCF Security Profiles values the Device achieved as part of OCF Conformance  
1887 testing. The DOTS may locate conformance results by inspecting manufacturer certificates  
1888 supplied with the OCF Device by selecting the "credusage" Property of the "/oic/sec/cred"  
1889 Resource having the value of "oic.sec.cred.mfgcert". The DOTS may further locate conformance  
1890 results by visiting a well-known OCF web site URI corresponding to the ocCPLAttributes  
1891 extension fields (clause 9.4.2.2.7). The DOTS may select a subset of Security Profiles (from  
1892 those evaluated by OCF conformance testing) based on a local policy.

1893 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries  
1894 to allow DOTS access subsequent to onboarding.

1895 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the  
1896 value that most correctly depicts the OCF Security Domain owner's intended Device deployment  
1897 strategy.

1898 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID")  
1899 to indicate the OCF Security Domain owner's intention to segment the OCF Security Domain  
1900 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the  
1901 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security  
1902 Profiles when issuing role credentials.

1903 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control  
1904 entries that include the role designation(s).

## 1905 **7.4 Provisioning**

### 1906 **7.4.1 Provisioning Flows**

#### 1907 **7.4.1.1 Provisioning Flows General**

1908 As part of onboarding a new Device a secure channel is formed between the new Device and the  
1909 OBT. Subsequent to the Device ownership status being changed to "owned", there is an  
1910 opportunity to begin provisioning. The OBT decides how the new Device will be managed going  
1911 forward and provisions the support services that should be subsequently used to complete  
1912 Device provisioning and on-going Device management.

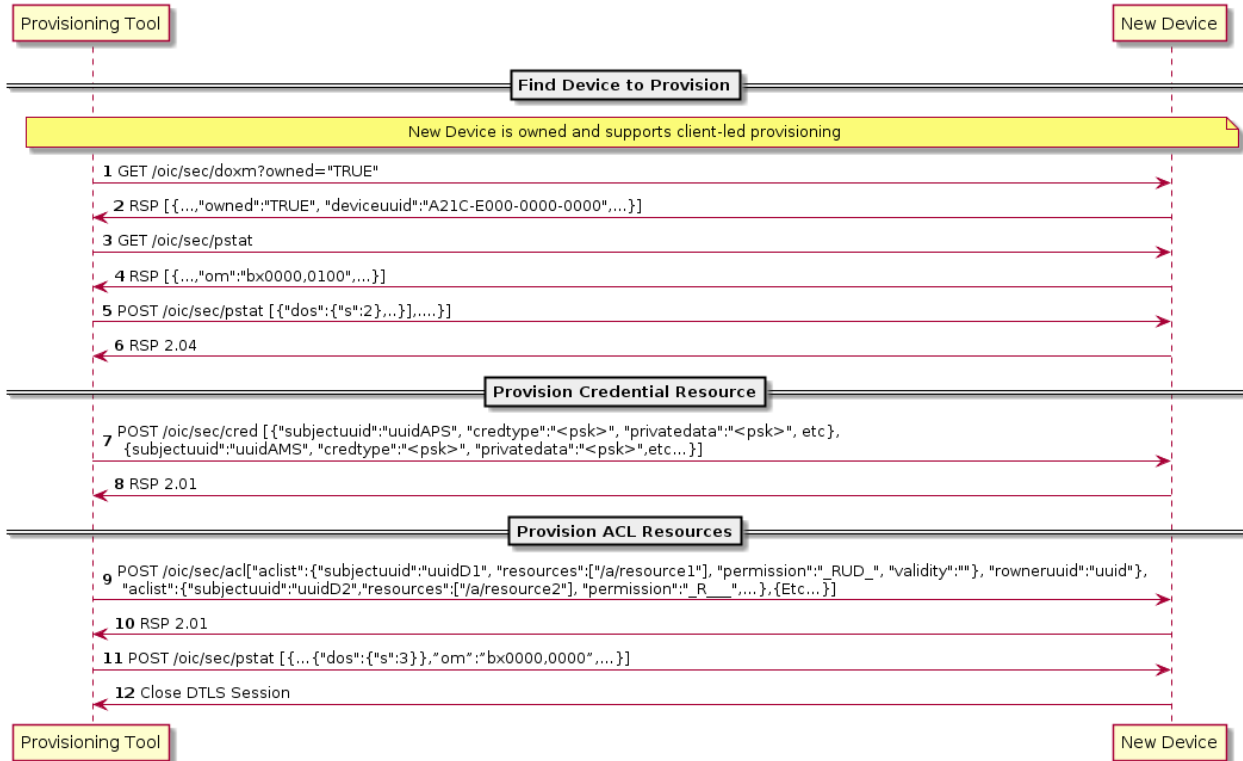
1913 The Device employs a Server-directed or Client-directed provisioning strategy. The  
1914 "/oic/sec/pstat" Resource identifies the provisioning strategy and current provisioning status. The  
1915 provisioning service should determine which provisioning strategy is most appropriate for the  
1916 OCF Security Domain. See 13.8 for additional detail.

#### 1917 **7.4.1.2 Client-directed Provisioning**

1918 Client-directed provisioning relies on a provisioning service that identifies Servers in need of  
1919 provisioning then performs all necessary provisioning duties.

1920 An example of Client-directed provisioning is shown in Figure 24 and steps described in Table 12.

OCF Client Led Provisioning  
with a Single Service Provider



1921  
1922  
1923

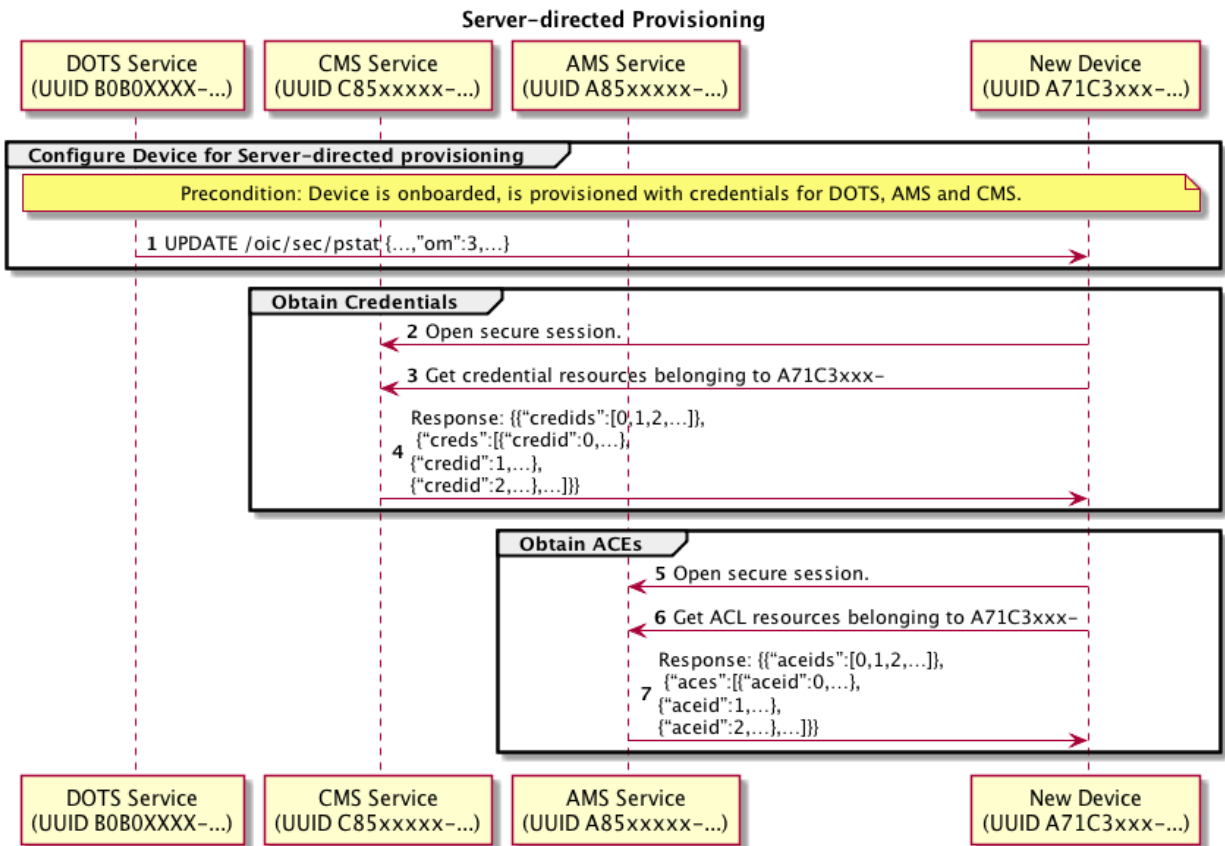
**Figure 24 – Example of Client-directed provisioning**  
**Table 12 – Steps describing Client -directed provisioning**

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/acl" 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.

1924 **7.4.1.3 Server-directed Provisioning**

1925 Server-directed provisioning relies on the Server (i.e. new Device) for directing much of the  
 1926 provisioning work. As part of the onboarding process the support services used by the Server to  
 1927 seek additional provisioning are provisioned. The new Device uses a self-directed, state-driven  
 1928 approach to analyse current provisioning state, and tries to drive toward target state. This  
 1929 example assumes a single support service is used to provision the new Device.

1930 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1931 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**

1932 **Table 13 – Steps for Server-directed provisioning using a single provisioning service**

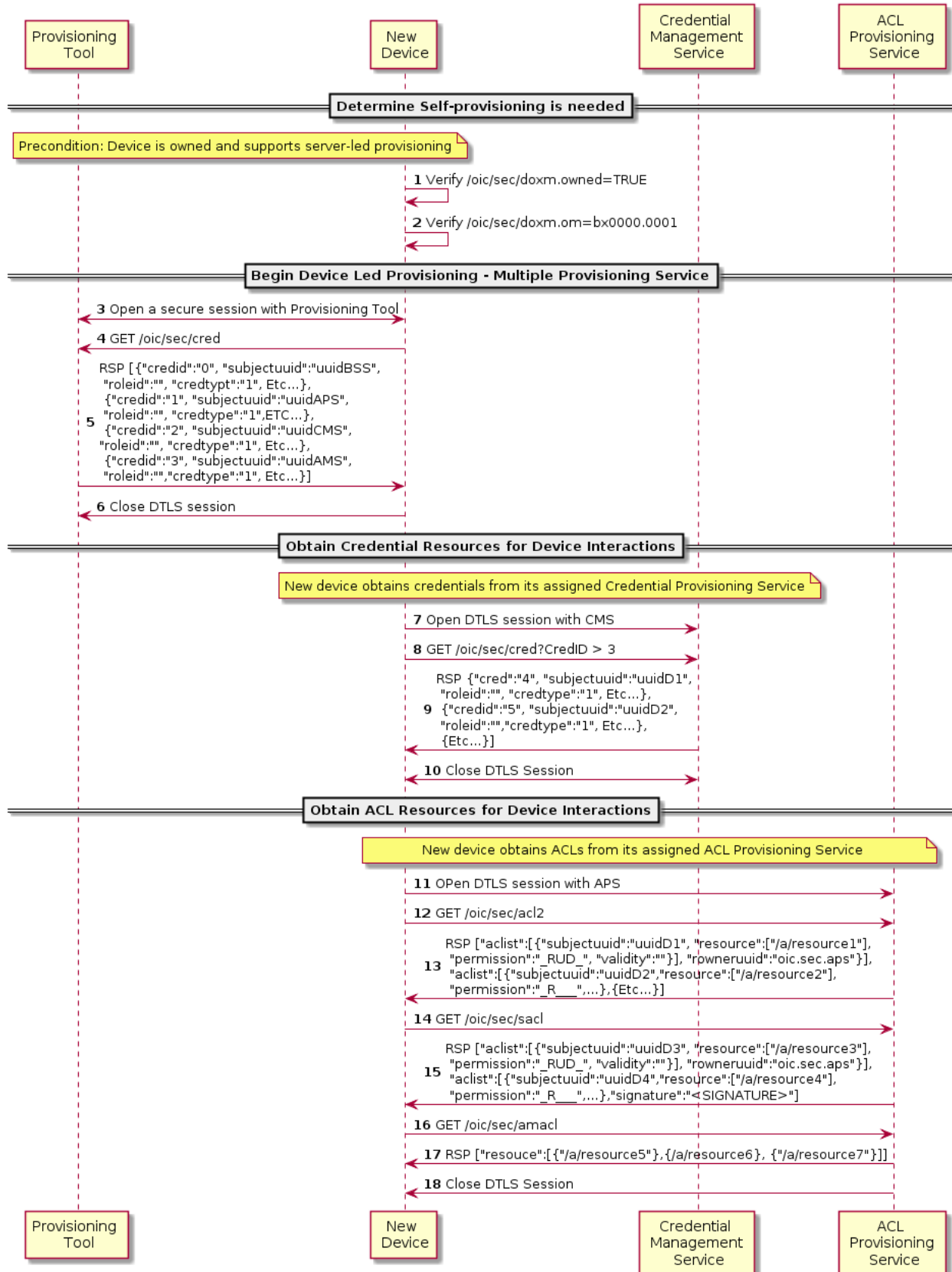
Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a secure session with the provisioning tool using the "/oic/sec/doxm". DevOwner value to open a TLS connection using SharedKey.
8 – 9	The new Devices 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/acl" Resources.
14	The secure session is closed.

1934 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1935 A Server-directed provisioning flow, involving multiple support services distributes the  
 1936 provisioning work across multiple support services. Employing multiple support services is an  
 1937 effective way to distribute provisioning workload or to deploy specialized support. The example in  
 1938 Figure 26 demonstrates using a provisioning tool to configure two support services, a CMS and  
 1939 an AMS. Steps for the example are described in Table 14.

### OCF Server Led Provisioning with Multiple Service Providers



1941 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1942 **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, rowneruuid 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, rowneruuid 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.

1943 **7.5 Device Provisioning for OCF Cloud**

1944 **7.5.1 Cloud Provisioning General**

1945 The Device that connects to the OCF Cloud shall support the "oic.r.coapcloudconf" Resource on  
1946 Device and following SVRs on the OCF Cloud: "/oic/sec/account", "/oic/sec/session",  
1947 "/oic/sec/tokenrefresh".

1948 The OCF Cloud is expected to use a secure mechanism for associating a Mediator with an OCF  
1949 Cloud User. The choice of mechanism is up to the OCF Cloud. Example, mechanisms include  
1950 HTTP authentication (with username and password) or OAuth 2.0 (using an Authorization Server  
1951 which could be operated by the OCF Cloud provider or a third party). OCF Cloud is expected to  
1952 ensure that the suitable authentication mechanism is used to authenticate the OCF Cloud User.

1953 **7.5.2 Device Provisioning by Mediator**

1954 The Mediator and the Device shall use the secure session to provision the Device to connect with  
1955 the OCF Cloud.

1956 The Mediator obtains an Access Token from the OCF Cloud as described in OCF Cloud  
1957 Specification. This Access Token is then used by the Device for registering with the OCF Cloud

1958 as described in 10.5. The OCF Cloud maintains a map where Access Token and Mediator  
 1959 provided Device ID are stored. At the time of Device Registration OCF Cloud validates the  
 1960 Access Token and associates the TLS session with corresponding Device ID.

1961 The Mediator provisions the Device, as described in OCF Cloud Specification. The Mediator  
 1962 provisions OCF Cloud URI to the "cis" Property of "oic.r.coapcloudconf" Resource, OCF Cloud  
 1963 UUID to the "sid" Property of "oic.r.coapcloudconf" Resource and per-device Access Token to the  
 1964 "at" Property of "oic.r.coapcloudconf" Resource on Device. Provisioned "at" is to be treated by  
 1965 Device as an Access Token with "Bearer" token type as defined in IETF RFC 6750.

1966 For the purposes of access control, the Device shall identify the OCF Cloud using the OCF Cloud  
 1967 UUID in the Common Name field of the End-Entity certificate used to authenticate the OCF Cloud.

1968 AMS should configure the ACE2 entries on a Device so that the Mediator(s) is the only Device(s)  
 1969 with UPDATE permission for the "oic.r.coapcloudconf" Resource.

1970 The AMS should configure the ACE2 entries on the Device to allow request from the OCF Cloud.  
 1971 By request from the Mediator, the AMS removes old ACL2 entries with previous OCF Cloud UUID.  
 1972 This request happens before "oic.r.coapcloudconf" is configured by the Mediator for the new OCF  
 1973 Cloud. The Mediator also requests AMS to set the OCF Cloud UUID as the "subject" Property for  
 1974 the new ACL2 entries. AMS may use "sid" Property of "oic.r.coapcloudconf" Resource as the  
 1975 current OCF Cloud UUID. AMS could either provision a wildcard entry for the OCF Cloud or  
 1976 provision an entry listing each Resource published on the Device.

1977 If OCF Cloud provides "redirecturi" Value as response during Device Registration, the redirected-  
 1978 to OCF Cloud is assumed to have the same OCF Cloud UUID and to use the same trust anchor.  
 1979 Otherwise, presented OCF Cloud UUID wouldn't match the provisioned ACL2 entries.

1980 The Mediator should provision the "oic.r.coapcloudconf" Resource with the Properties in Table 15.  
 1981 These details once provisioned are used by the Device to perform Device Registration to the  
 1982 OCF Cloud. After the initial registration, the Device should use updated values received from the  
 1983 OCF Cloud instead. If OCF Cloud User wants the Device to re-register with the OCF Cloud, they  
 1984 can use the Mediator to re-provision the "oic.r.coapcloudconf" Resource with the new values.

**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.

## 1987 **8 Device Onboarding State Definitions**

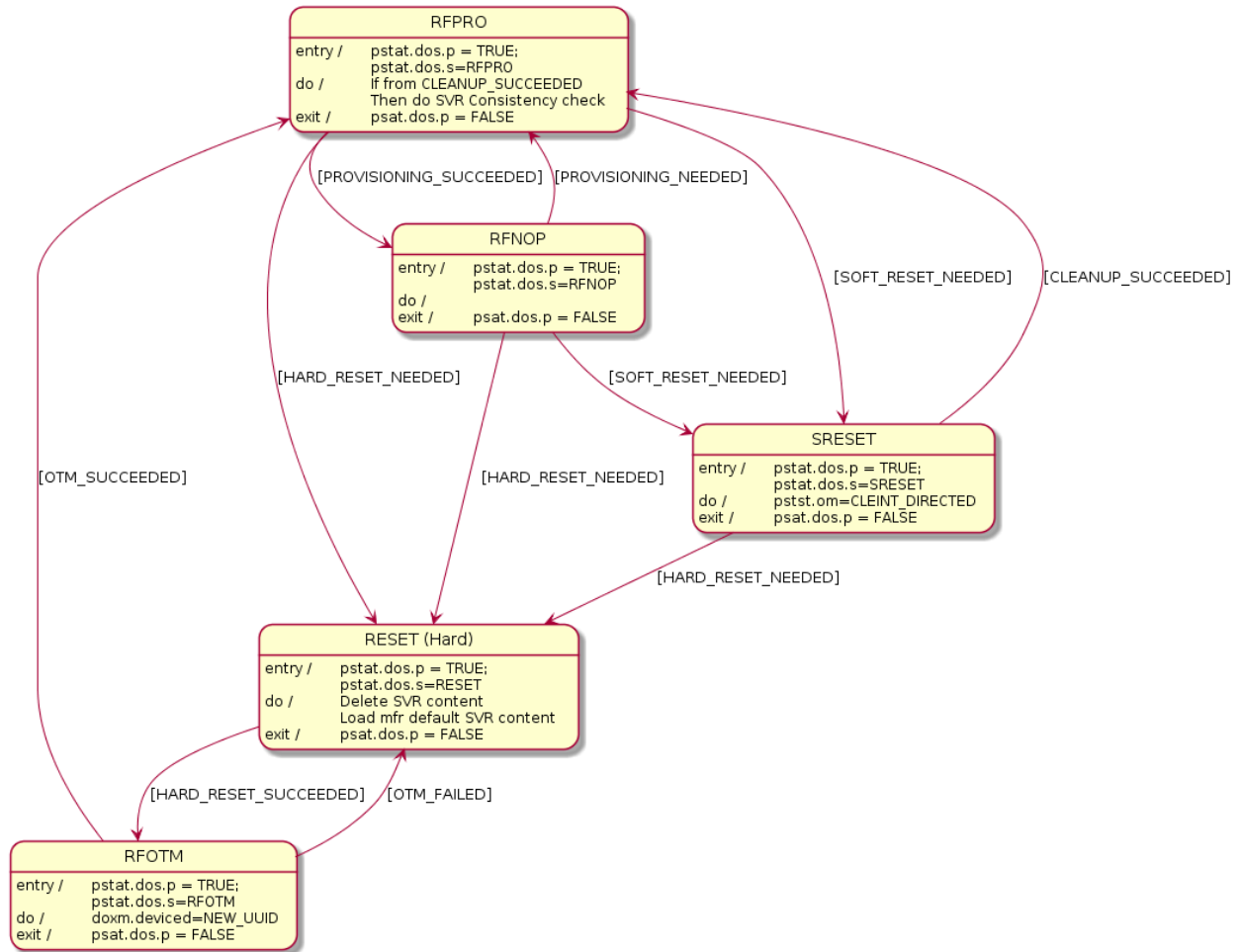
### 1988 **8.1 Device Onboarding General**

1989 As explained in 5.3, the process of onboarding completes after the ownership of the Device has  
 1990 been transferred and the Device has been provisioned with relevant configuration/services as



1991 explained in 5.4. The Figure 27 shows the various states a Device can be in during the Device  
 1992 lifecycle.

1993 The /pstat.dos.s Property is RW by the /oic/sec/pstat resource owner (e.g. "doxs" service) so that  
 1994 the resource owner can remotely update the Device state. When the Device is in RFNOP or  
 1995 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the  
 1996 Device state is SRESET the Device OC may be the only indication of authorization to access the  
 1997 Device. The Device owner may perform low-level consistency checks and re-provisioning to get  
 1998 the Device suitable for a transition to RFPRO.



1999 **Figure 27 – Device state model**

2001 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the  
 2002 "Ready for Normal Operation" state where it has all it needs in order to start interoperating with  
 2003 other Devices. 8.2 specifies the minimum mandatory configuration that a Device shall hold in  
 2004 order to be considered as "Ready for Normal Operation".

2005 In the event of power loss or Device failure, the Device should remain in the same state that it  
 2006 was in prior to the power loss / failure

2007 If a Device or resource owner OBSERVEs /pstat.dos.s, then transitions to SRESET will give early  
 2008 warning notification of Devices that may require SVR consistency checking.

2009 In order for onboarding to function, the Device shall have the following Resources installed:

- 2010 1) "/oic/sec/doxm" Resource  
2011 2) "/oic/sec/pstat" Resource  
2012 3) "/oic/sec/cred" Resource  
2013 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5  
2014 and 8.6.

## 2015 **8.2 Device Onboarding-Reset State Definition**

2016 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset  
2017 also defines a state where the Device asset is ready to be transferred to another party.

2018 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces  
2019 Platform reset. All Devices hosted on the same Platform transition their Device states to RESET  
2020 when the Platform reset is asserted.

2021 The following Resources and their specific properties shall have the value as specified:

- 2022 1) The owned Property of the "/oic/sec/doxm" Resource shall transition to FALSE.  
2023 2) The devowneruuid Property of the "/oic/sec/doxm" Resource shall be nil UUID.  
2024 3) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2025 implemented.  
2026 4) The deviceuuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2027 default value.  
2028 5) The deviceid Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2029 default value, if this Property is implemented.  
2030 6) The sct Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's default  
2031 value.  
2032 7) The oxmsel Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's  
2033 default value.  
2034 8) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.  
2035 9) The dos Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal  
2036 "RESET" state and dos.p shall equal "FALSE".  
2037 10)  
2038 11) The om (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the  
2039 manufacturer default value.  
2040 12) The sm (supported operational modes) Property of the "/oic/sec/pstat" Resource shall be set  
2041 to the manufacturer default value.  
2042 13) The rowneruuid Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl", "/oic/sec/amacl",  
2043 "/oic/sec/sacl", and "/oic/sec/cred" Resources shall be nil UUID.  
2044 14) The supportedprofiles Property of the "/oic/sec/sp" Resource shall be set to the manufacturer  
2045 default value.  
2046 15) The currentprofile Property of the "/oic/sec/sp" Resource shall be set to the manufacturer  
2047 default value.

## 2048 **8.3 Device Ready-for-OTM State Definition**

2049 The following Resources and their specific properties shall have the value as specified when the  
2050 Device enters ready for ownership transfer:

- 2051 1) The owned Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to  
2052 TRUE.

- 2053 2) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is  
2054 implemented.
- 2055 3) The devowneruuid Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2056 4) The deviceid Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is  
2057 implemented. The value of the di Property in "/oic/d" is undefined.
- 2058 5) The deviceuuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer  
2059 default value.
- 2060 6) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2061 7) The dos of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RFOTM" state  
2062 and dos.p shall equal "FALSE".
- 2063 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

#### 2064 **8.4 Device Ready-for-Provisioning State Definition**

2065 The following Resources and their specific properties shall have the value as specified when the  
2066 Device enters ready for provisioning:

- 2067 1) The owned Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2068 2) The devowneruuid Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2069 3) The deviceuuid Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
2070 set to the value that was determined during RFOTM processing. Also the value of the di  
2071 Property in "/oic/d" Resource shall be the same as the deviceid Property in the  
2072 "/oic/sec/doxm" Resource.
- 2073 4) The oxmsel Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM  
2074 used during ownership transfer.
- 2075 5) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2076 6) The dos of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RFPRO" state  
2077 and dos.p shall equal "FALSE".
- 2078 7) The rowneruuid Property of every installed Resource shall be set to a valid Resource owner  
2079 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2080 rowneruuid may result in an orphan Resource.
- 2081 8) The "/oic/sec/cred" Resource shall contain credentials for each entity referenced by an  
2082 rowneruuid, amsuuid, devowneruuid.

#### 2083 **8.5 Device Ready-for-Normal-Operation State Definition**

2084 The following Resources and their specific properties shall have the value as specified when the  
2085 Device enters ready for normal operation:

- 2086 1) The owned Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2087 2) The devowneruuid Property of the "/oic/sec/doxm" Resource shall not be nil UUID.
- 2088 3) The deviceuuid Property of the "/oic/sec/doxm" Resource shall not be nil UUID and shall be  
2089 set to the ID that was configured during OTM. Also the value of the "di" Property in "/oic/d"  
2090 shall be the same as the deviceuuid.
- 2091 4) The oxmsel Property of the "/oic/sec/doxm" Resource shall have the value of the actual OTM  
2092 used during ownership transfer.
- 2093 5) The isop Property of the "/oic/sec/pstat" Resource shall be set to TRUE by the Server once  
2094 transition to RFNOP is otherwise complete.
- 2095 6) The dos of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RFNOP" state  
2096 and dos.p shall equal "FALSE".

2097 7) The rowneruuid Property of every installed Resource shall be set to a valid resource owner  
2098 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a  
2099 rowneruuid results in an orphan Resource.

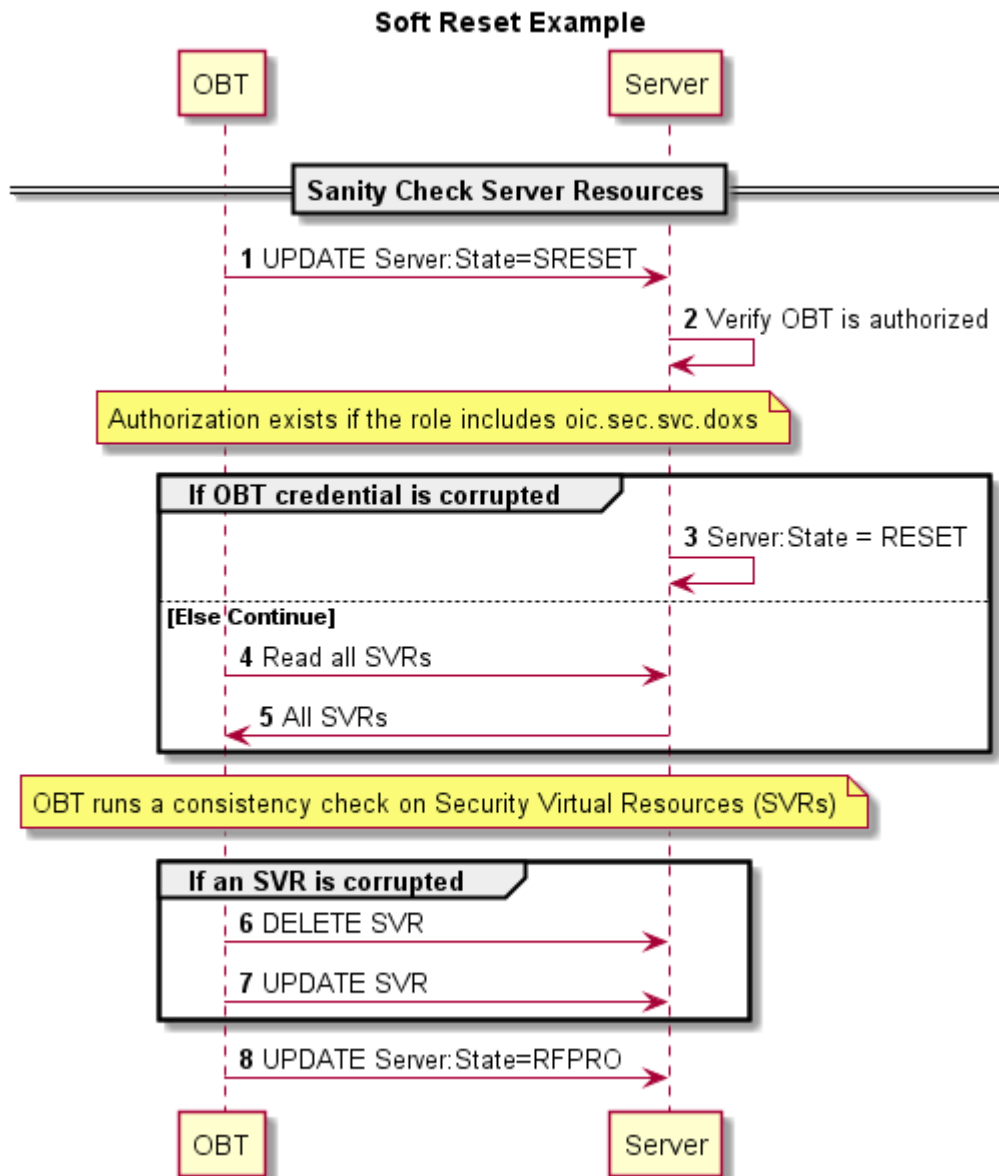
2100 8) The "/oic/sec/cred" Resource shall contain credentials for each service referenced by a  
2101 rowneruuid, amsuuid, devowneruuid.

## 2102 **8.6 Device Soft Reset State Definition**

2103 The soft reset state is defined (e.g. /pstat.dos.s = SRESET) where entrance into this state means  
2104 the Device is not operational but remains owned by the current owner. The Device may exit  
2105 SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during  
2106 original onboarding (but should not require use of an OTM /doxm.oxms).

2107 The DOTS should perform a consistency check of the SVR and if necessary, re-provision them  
2108 sufficiently to allow the Device to transition to RFPRO.

2109 Figure 28 depicts OBT Sanity Check Sequence in SRESET.



2110  
2111

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

2112 The DOTS should perform a sanity check of SVRs before final transition to RFPRO Device state.  
2113 If the DOTS credential cannot be found or is determined to be corrupted, the Device state  
2114 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to  
2115 validate the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS  
2116 Devices.

2117 When in SRESET, the following Resources and their specific Properties shall have the values as  
2118 specified.

- 2119 1) The owned Property of the "/oic/sec/doxm" Resource shall be TRUE.
- 2120 2) The devowneruuid Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2121 3) The devowner Property of the "/oic/sec/doxm" Resource shall be non-null, if this Property is  
2122 implemented.

- 2123 4) The deviceuuidProperty of the "/oic/sec/doxm" Resource shall remain non-null.
- 2124 5) The deviceid Property of the "/oic/sec/doxm" Resource shall remain non-null.
- 2125 6) The sct Property of the "/oic/sec/doxm" Resource shall retain its value.
- 2126 7) The oxmsel Property of the "/oic/sec/doxm" Resource shall retains its value.
- 2127 8) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.
- 2128 9) The "/oic/sec/pstat.dos.s" Property shall be SRESET.
- 2129 10)The om (operational modes) Property of the "/oic/sec/pstat" Resource shall be "client-directed  
2130 mode".
- 2131 11)The sm (supported operational modes) Property of "/oic/sec/pstat" Resource may be updated  
2132 by the Device owner (aka DOTS).
- 2133 12)The rowneruuid Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl", "/oic/sec/acl2",  
2134 "/oic/sec/amacl", "/oic/sec/sacl", and "/oic/sec/cred" Resources may be reset by the Device  
2135 owner (aka DOTS) and re-provisioned.
- 2136

2137 **9 Security Credential Management**

2138 **9.1 Preamble**

2139 This clause provides an overview of the credential types in OCF, along with details of credential  
2140 use, provisioning and ongoing management.

2141 **9.2 Credential Lifecycle**

2142 **9.2.1 Credential Lifecycle General**

2143 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)  
2144 issuance and (5) revocation.

2145 **9.2.2 Creation**

2146 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is  
2147 authorized by matching the rowneruuid Property of the "/oic/sec/cred" resource to the DeviceID of  
2148 the credential the CMS used to establish the secure connection.

2149 Credential Resources created using a CMS may involve specialized credential issuance protocols  
2150 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate  
2151 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of  
2152 a provisioning action by a DOTS, CMS or AMS.

2153 **9.2.3 Deletion**

2154 The CMS should delete known compromised credential Resources. The Device (e.g. the Device  
2155 where the credential Resource is hosted) should delete credential Resources that have expired.

2156 An expired credential Resource may be deleted to manage memory and storage space.

2157 Deletion in OCF key management is equivalent to credential suspension.

2158 **9.2.4 Refresh**

2159 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2160 The "/oic/sec/cred" Resource supports expiry using the Period Property. Credential refresh may  
2161 be applied when a credential is about to expire or is about to exceed a maximum threshold for  
2162 bytes encrypted.

2163 A credential refresh method specifies the options available when performing key refresh. The  
2164 Period Property informs when the credential should expire. The Device may proactively obtain a  
2165 new credential using a credential refresh method using current unexpired credentials to refresh  
2166 the existing credential. If the Device does not have an internal time source, the current time  
2167 should be obtained from a CMS at regular intervals.

2168 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the  
2169 CMS credentials to the Device. If the onboarding established credentials are allowed to expire  
2170 the DOTS shall re-onboard the Device to re-apply device owner transfer steps.

2171 All Devices shall support at least one credential refresh method.

2172 **9.2.5 Revocation**

2173 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where  
2174 the revocation method involves provisioning of a revocation object that identifies a credential that  
2175 is to be revoked prior to its normal expiration period, a credential Resource is created containing  
2176 the revocation information that supersedes the originally issued credential. The revocation object

2177 expiration should match that of the revoked credential so that the revocation object is cleaned up  
2178 upon expiry.

2179 It is conceptually reasonable to consider revocation applying to a credential or to a Device.  
2180 Device revocation asserts all credentials associated with the revoked Device should be  
2181 considered for revocation. Device revocation is necessary when a Device is lost, stolen or  
2182 compromised. Deletion of credentials on a revoked Device might not be possible or reliable.

### 2183 **9.3 Credential Types**

#### 2184 **9.3.1 Preamble**

2185 The "/oic/sec/cred" Resource maintains a credential type Property that supports several  
2186 cryptographic keys and other information used for authentication and data protection. The  
2187 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric  
2188 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.  
2189 PIN/password).

#### 2190 **9.3.2 Pair-wise Symmetric Key Credentials**

2191 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The  
2192 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2193 Pair-wise keys could be established through ad-hoc key agreement protocols.

2194 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2195 The PublicData Property may contain a token encrypted to the peer Device containing the pair-  
2196 wise key.

2197 The OptionalData Property may contain revocation status.

2198 The Device implementer should apply hardened key storage techniques that ensure the  
2199 PrivateData remains private.

2200 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2201 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2202 unauthorized modifications.

#### 2203 **9.3.3 Group Symmetric Key Credentials**

2204 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are  
2205 used for efficient sharing of data among group participants.

2206 Group keys do not provide authentication of Devices but only establish membership in a group.

2207 The CMS shall provision group symmetric key credentials to the group members. The CMS  
2208 maintains the group memberships.

2209 The PrivateData Property in the "/oic/sec/cred" Resource contains the symmetric key.

2210 The PublicData Property may contain the group name.

2211 The OptionalData Property may contain revocation status.

2212 The Device implementer should apply hardened key storage techniques that ensure the  
2213 PrivateData remains private.



2214 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2215 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2216 unauthorized modifications.

### 2217 **9.3.4 Asymmetric Authentication Key Credentials**

#### 2218 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2219 Asymmetric authentication key credentials contain either a public and private key pair or only a  
2220 public key. The private key is used to sign Device authentication challenges. The public key is  
2221 used to verify a device authentication challenge-response.

2222 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2223 The PublicData Property contains the public key.

2224 The OptionalData Property may contain revocation status.

2225 The Device implementer should apply hardened key storage techniques that ensure the  
2226 PrivateData remains private.

2227 Devices should generate asymmetric authentication key pairs internally to ensure the private key  
2228 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material  
2229 between Devices.

2230 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2231 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2232 unauthorized modifications.

#### 2233 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2234 Devices should employ industry-standard high-assurance techniques when allowing off-device  
2235 key pair creation and provisioning. Use of such key pairs should be minimized, particularly if the  
2236 key pair is immutable and cannot be changed or replaced after provisioning.

2237 When used as part of onboarding, these key pairs can be used to prove the Device possesses  
2238 the manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept  
2239 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the  
2240 Device, and then provisions new OCF Security Domain credentials for use.

### 2241 **9.3.5 Asymmetric Key Encryption Key Credentials**

2242 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when  
2243 distributing or storing the key.

2244 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2245 The PublicData Property contains the public key.

2246 The OptionalData Property may contain revocation status.

2247 The Device implementer should apply hardened key storage techniques that ensure the  
2248 PrivateData remains private.

2249 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2250 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2251 unauthorized modifications.

2252 **9.3.6 Certificate Credentials**

2253 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a  
2254 CMS or an external certificate authority (CA).

2255 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2256 The issued certificate is stored with the asymmetric key credential Resource.

2257 Other objects useful in managing certificate lifecycle such as certificate revocation status are  
2258 associated with the credential Resource.

2259 Either an asymmetric key credential Resource or a self-signed certificate credential is used to  
2260 terminate a path validation.

2261 The PrivateData Property in the "/oic/sec/cred" Resource contains the private key.

2262 The PublicData Property contains the issued certificate.

2263 The OptionalData Property may contain revocation status.

2264 The Device implementer should apply hardened key storage techniques that ensure the  
2265 PrivateData remains private.

2266 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2267 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2268 unauthorized modifications.

2269 **9.3.7 Password Credentials**

2270 Shared secret credentials are used to maintain a PIN or password that authorizes Device access  
2271 to a foreign system or Device that doesn't support any other OCF credential types.

2272 The PrivateData Property in the "/oic/sec/cred" Resource contains the PIN, password and other  
2273 values useful for changing and verifying the password.

2274 The PublicData Property may contain the user or account name if applicable.

2275 The OptionalData Property may contain revocation status.

2276 The Device implementer should apply hardened key storage techniques that ensure the  
2277 PrivateData remains private.

2278 The Device implementer should apply appropriate integrity, confidentiality and access protection  
2279 of the "/oic/sec/cred", "/oic/sec/crl", "/oic/sec/roles", "/oic/sec/csr" Resources to prevent  
2280 unauthorized modifications.

2281 **9.4 Certificate Based Key Management**

2282 **9.4.1 Overview**

2283 To achieve authentication and transport security during communications in OCF Security Domain,  
2284 certificates containing public keys of communicating parties and private keys can be used.

2285 The certificate and private key may be issued by a local or remote certificate authority (CA). For  
2286 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of  
2287 identity. In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to  
2288 validate proof of identity and validity.

2289 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,  
2290 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in  
2291 X.509 are not supported so that the format intends to meet the constrained Device's requirement.

2292 As for the certificate and CRL management in the Server, the process of storing, retrieving and  
2293 parsing Resources of the certificates and CRL will be performed at the security resource  
2294 manager layer; the relevant interfaces may be exposed to the upper layer.

2295 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of  
2296 certificates and CRL will be stored and managed in SVR database.

2297 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a  
2298 CMS to a Device when it is newly onboarded. The issuing CMS should process certificate  
2299 revocations for certificates it issues. If a certificate private key is compromised, the CMS should  
2300 revoke the certificate. If CRLs are used by a Device, the CMS should regularly (for example;  
2301 every 3 months) update the "/oic/sec/crl" resource for the Devices it manages.

## 2302 **9.4.2 X.509 Digital Certificate Profiles**

### 2303 **9.4.2.1 Digital Certificate Profile General**

2304 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in  
2305 IETF RFC 5280.

2306 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications  
2307 for those communities wishing to make use of X.509 technology. The X.509 v3 certificate format  
2308 is described in detail, with additional information regarding the format and semantics of OCF  
2309 specific extension(s). The supported standard certificate extensions are also listed.

2310 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this  
2311 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are  
2312 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,  
2313 compliant entities shall ignore their contents.

2314 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as  
2315 defined in ISO/IEC 8825-1 to encode certificates.

2316 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public  
2317 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs  
2318 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and  
2319 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2320 The following clauses specify the supported standard and custom extensions for the OCF  
2321 certificates profile.

### 2322 **9.4.2.2 Certificate Profile and Fields**

#### 2323 **9.4.2.2.1 Root CA Certificate Profile**

2324 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2325 **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)

2326 Table 17 describes X.509 v3 extensions required for Root CA Certificates.

2327 **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)

2328 **9.4.2.2.2 Intermediate CA Certificate Profile**

2329 Table 18 describes X.509 v1 fields required for Intermediate CA Certificates.

2330 **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)

2331 Table 19 **describes** X.509 v3 extensions required for Intermediate CA Certificates.

**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

2333 **9.4.2.2.3 End-Entity Black Certificate Profile**2334 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security  
2335 profile.

2336

**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)

2337 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

2339 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2340 The OCF Compliance Extension defines required parameters to correctly identify the type of  
2341 Device, its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2342 The extension carries an "ocfVersion" field which provides the specific base version of the OCF  
2343 documents the device implements. The "ocfVersion" field shall contain a sequence of three  
2344 integers ("major", "minor", and "build"). For example, if an entity is certified to be compliant with

2345 OCF specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be  
2346 set to "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote  
2347 compliance to a specified base version of the OCF documents.

2348 The "securityProfile" field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more  
2349 supported Security Profiles associated with the certificate in string form (UTF-8). All Security  
2350 Profiles associated with the certificate should be identified by this field.

2351 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".  
2352 The fields carry human-readable descriptions of the Device's name and manufacturer,  
2353 respectively.

2354 The ASN.1 definition of the OCFCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined  
2355 as follows:

```
2356 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2357                               private(4) enterprise(1) OCF(51414) }
2358
2359 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2360
2361 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2362
2363 ocfVersion ::= SEQUENCE {
2364     major INTEGER,
2365         --Major version number
2366     minor INTEGER,
2367         --Minor version number
2368     build INTEGER,
2369         --Build/Micro version number
2370 }
2371
2372 ocfCompliance ::= SEQUENCE {
2373     version ocfVersion,
2374         --Device/OCF version
2375     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2376         --Sequence of OCF Security Profile OID strings
2377         --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2378     deviceName UTF8String,
2379         --Name of the device
2380     deviceManufacturer UTF8String,
2381         --Human-Readable Manufacturer
2382         --of the device
2383 }
```

#### 2384 9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension

2385 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for  
2386 devices to signal to the network the access and network functionality they require to properly  
2387 function. Access controls can be more easily achieved and deployed at scale when the MUD  
2388 extension is used. The current draft of the MUD v3 extension at this time of writing is:

2389 <https://tools.ietf.org/html/rfc8520#section-11>

2390 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2391 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2392                       internet(1) security(5) mechanisms(5) pkix(7)
2393                       id-mod(0) id-mod-mudURLExtn2016(88) }
2394
2395 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2396 -- EXPORTS ALL --
2397 IMPORTS
```



```

2398     EXTENSION
2399     FROM PKIX-CommonTypes-2009
2400         { iso(1) identified-organization(3) dod(6) internet(1)
2401           security(5) mechanisms(5) pkix(7) id-mod(0)
2402           id-mod-pkixCommon-02(57) }
2403     id-pe
2404     FROM PKIX1Explicit-2009
2405         { iso(1) identified-organization(3) dod(6) internet(1)
2406           security(5) mechanisms(5) pkix(7) id-mod(0)
2407           id-mod-pkix1-explicit-02(51) } ;
2408     MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2409     ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2410                               IDENTIFIED BY id-pe-mud-url }
2411
2412     id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2413
2414     MUDURLSyntax ::= IA5String
2415
2416     END

```

#### 2417 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2418 The OCF Security Claims Extension defines a list of OIDs representing security claims that the  
2419 manufacturer/integrator is making as to the security posture of the device above those required  
2420 by the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2421 The purpose of this extension is to allow for programmatic evaluation of assertions made about  
2422 security to enable some platforms/policies/administrators to better understand what is being  
2423 onboarded or challenged.

2424 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is  
2425 defined as follows:

```

2426 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2427                               private(4) enterprise(1) OCF(51414) }
2428
2429     id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2430
2431     id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2432
2433     claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2434     --Device claims that the boot process follows a procedure trusted
2435     --by the firmware and the BIOS
2436
2437     claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2438     --Device claims that credentials are stored in a specialized hardware
2439     --protection environment such as a Trusted Platform Module (TPM) or
2440     --similar mechanism.
2441
2442     ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2443
2444     ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

#### 2445 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2446 The OCF Certified Product List Extension defines required parameters to utilize the OCF  
2447 Compliance Management System Certified Product List (OCMS-CPL). This clause is only  
2448 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to  
2449 verify the compliance level of a device.

2450 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model  
2451 and Version.

2452 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique  
2453 PEN established in the IANA PEN list located at: [https://www.iana.org/assignments/enterprise-](https://www.iana.org/assignments/enterprise-numbers)  
2454 [numbers](https://www.iana.org/assignments/enterprise-numbers). The 'cpl-at-IANAPen' field found in end-products shall be the same information as  
2455 reported during OCF Certification.

2456 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field  
2457 found in end-products shall be the same information as reported during OCF Certification.

2458 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found  
2459 in end-products shall be the same information as reported during OCF Certification.

2460 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is  
2461 defined as follows:

```
2462 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2463                               private(4) enterprise(1) OCF(51414) }
2464
2465 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2466
2467     id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2468
2469     cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2470     cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2471     cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2472
2473
2474     ocfCPLAttributes ::= SEQUENCE {
2475         cpl-at-IANAPen      UTF8String,
2476         --Manufacturer's registered IANA Private Enterprise Number
2477         cpl-at-model       UTF8String,
2478         --Device OCF Security Profile
2479         cpl-at-version     UTF8String
2480         --Name of the device
2481     }
```

### 2482 9.4.2.3 Supported Certificate Extensions

2483 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the  
2484 clause number from that RFC to include it by reference. Each extension is summarized here, and  
2485 any modifications to the RFC definition are listed. Devices MUST implement and understand the  
2486 extensions listed here; other extensions from the RFC are not included in this document and  
2487 therefore are not required. 10.4 describes what Devices must implement when validating  
2488 certificate chains, including processing of extensions, and actions to take when certain  
2489 extensions are absent.

#### 2490 – Authority Key Identifier (4.2.1.1)

2491 The Authority Key Identifier (AKI) extension provides a means of identifying the public key  
2492 corresponding to the private key used to sign a certificate. This document makes the following  
2493 modifications to the referenced definition of this extension:

2494 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier  
2495 sequence are not permitted; only keyIdentifier is allowed. This results in the following  
2496 grammar definition:

```
2497 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2498
2499 AuthorityKeyIdentifier ::= SEQUENCE {
2500     keyIdentifier          [0] KeyIdentifier
2501 }
2502 KeyIdentifier ::= OCTET STRING
```

#### 2503 – Subject Key Identifier (4.2.1.2)

2504 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that  
2505 contain a particular public key.

2506 This document makes the following modification to the referenced definition of this extension:

2507 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's  
2508 SubjectPublicKeyInfo field or a method that generates unique values. This document  
2509 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey  
2510 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains  
2511 must not assume any particular method of computing key identifiers, however, and must only  
2512 base matching AKI's and SKI's in certification path constructions on key identifiers seen in  
2513 certificates.

#### 2514 – Subject Alternative Name

2515 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role  
2516 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and  
2517 interpreted as described below. When no EKU is present, or has another value, the  
2518 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to  
2519 encode one or more Role ID values in role certificates, binding the roles to the subject public  
2520 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2521 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2522
2523 SubjectAltName ::= GeneralNames
2524
2525 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2526
2527 GeneralName ::= CHOICE {
2528     otherName                [0]     OtherName,
2529     rfc5322Name              [1]     IA5String,
2530     dNSName                   [2]     IA5String,
2531     x400Address               [3]     ORAddress,
2532     directoryName             [4]     Name,
2533     ediPartyName              [5]     EDIPartyName,
2534     uniformResourceIdentifier [6]     IA5String,
2535     iPAddress                 [7]     OCTET STRING,
2536     registeredID              [8]     OBJECT IDENTIFIER }
2537
2538     EDIPartyName ::= SEQUENCE {
2539         nameAssigner          [0]     DirectoryString OPTIONAL,
2540         partyName              [1]     DirectoryString }
```

2541  
2542 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a  
2543 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name  
2544 shall contain exactly one CN (Common Name) component, and zero or one OU  
2545 (Organizational Unit) components. The OU component, if present, shall specify the authority  
2546 that defined the semantics of the role. If the OU component is absent, the certificate issuer  
2547 has defined the role. The CN component shall encode the role ID. Other GeneralName types  
2548 in the SEQUENCE may be present, but shall not be interpreted as roles. Therefore, if the  
2549 certificate issuer includes non-role names in the subjectAltName extension, the extension  
2550 should not be marked critical.

2551 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted  
2552 character set [0-9a-z-A-z '()+, -./:=?].

#### 2553 – Key Usage (4.2.1.3)

2554 The key usage extension defines the purpose (e.g., encipherment, signature, certificate  
2555 signing) of the key contained in the certificate. The usage restriction might be employed when  
2556 a key that could be used for more than one operation is to be restricted.

2557 This document does not modify the referenced definition of this extension.

2558 – Basic Constraints (4.2.1.9)

2559 The basic constraints extension identifies whether the subject of the certificate is a CA and  
2560 the maximum depth of valid certification paths that include this certificate. Without this  
2561 extension, a certificate cannot be an issuer of other certificates.

2562 This document does not modify the referenced definition of this extension.

2563 – Extended Key Usage (4.2.1.12)

2564 Extended Key Usage describes allowed purposes for which the certified public key may can  
2565 be used. When a Device receives a certificate, it determines the purpose based on the  
2566 context of the interaction in which the certificate is presented, and verifies the certificate can  
2567 be used for that purpose.

2568

2569 This document makes the following modifications to the referenced definition of this extension:  
2570 CAs SHOULD mark this extension as critical.

2571 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).  
2572

2573 The list of OCF-specific purposes and the assigned OIDs to represent them are:

2574 – Identity certificate 1.3.6.1.4.1.44924.1.6  
2575 – Role certificate 1.3.6.1.4.1.44924.1.7

2576 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**

2577 See 9.4.3.5 for details.

2578 **9.4.2.5 Encoding of Certificate**

2579 See 9.4.2 for details.

2580 **9.4.3 Certificate Revocation List (CRL) Profile**

2581 **9.4.3.1 CRL General**

2582 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use  
2583 within OCF applications for those communities wishing to support revocation features in their  
2584 PKIs.

2585 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in  
2586 IETF RFC 5280 – Clause 5.1

2587 **9.4.3.2 CRL Profile and Fields**

2588 This clause intentionally left empty.

2589 **9.4.3.3 Encoding of CRL**

2590 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]  
2591 should be used to encode CRL.

2592 **9.4.3.4 CRLs Supported Standard Extensions**

2593 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55]  
2594 provide methods for associating additional attributes with CRLs. The following list of X.509  
2595 extensions should be supported in this certificate profile:

2596 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides  
2597 a means of identifying the public key corresponding to the private key used to sign a CRL.  
2598 Conforming CRL issuers should use the key identifier method, and shall include this extension  
2599 in all CRLs issued

2600 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that  
2601 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer  
2602 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for  
2603 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]  
2604 [X9.55]. Although this document does not provide any recommendation about the use of specific  
2605 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not  
2606 marked critical.

#### 2607 **9.4.3.5 Encryption Ciphers and TLS support**

2608 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support  
2609 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 cipher suite as defined in IETF RFC 7251 and  
2610 may support additional ciphers as defined in the TLS v1.2 specifications.

#### 2611 **9.4.4 Resource Model**

2612 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated  
2613 with a separate crl Resource that is defined for maintaining the revocation list.

2614 The cred Resource contains the certificate information pertaining to the Device. The PublicData  
2615 Property holds the device certificate and CA certificate chain. PrivateData Property holds the  
2616 Device private key paired to the certificate. (See 13.3 for additional detail regarding the  
2617 "/oic/sec/cred" Resource).

2618 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained  
2619 through the CMS. The Device must consider revoked certificates as part of certificate path  
2620 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain  
2621 a full list, the Device must query the CMS for current revocation status. (See 13.4 for additional  
2622 detail regarding the "/oic/sec/crl" Resource).

#### 2623 **9.4.5 Certificate Provisioning**

2624 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have  
2625 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)  
2626 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate  
2627 shall have the format described in 9.4.2.

2628 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,  
2629 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer  
2630 them to the Device including its CA certificate chain. Optionally, the CMS may also transfer one  
2631 or more role certificates, which shall have the format described in clause 9.4.2. . The  
2632 subjectPublicKey of each role certificate shall match the subjectPublicKey in the Device  
2633 certificate.

2634 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in  
2635 IETF RFC 2986, and is included here by reference.

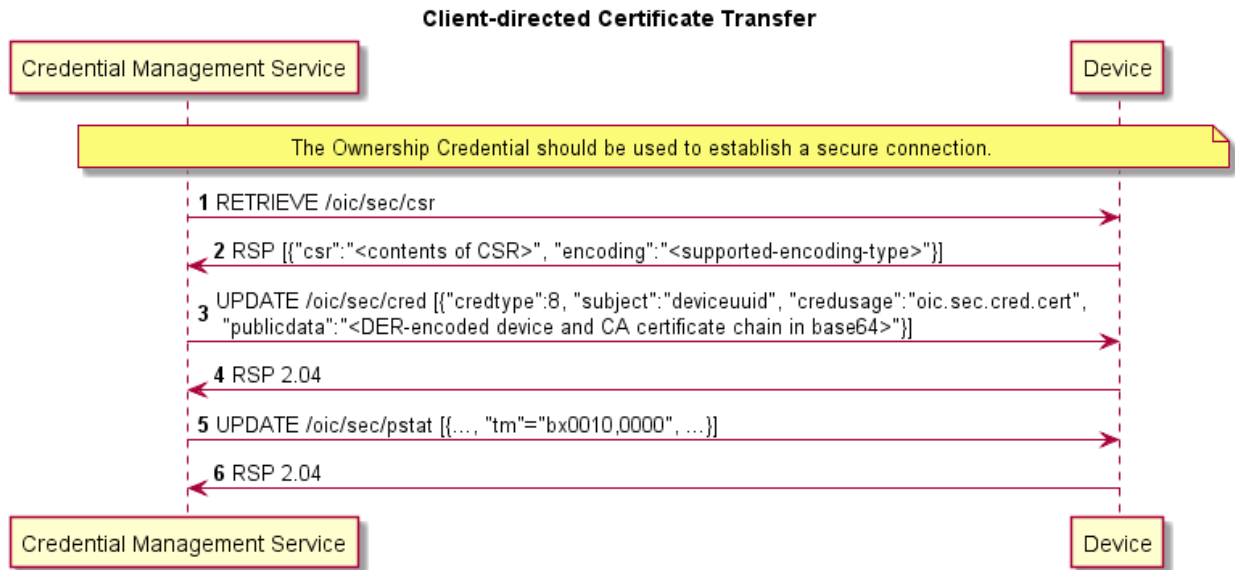
2636 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2637 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device  
2638 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.  
2639 The Device determines the public key to present; this may be an already-provisioned key it  
2640 has selected for use with authentication, or if none is present, it may generate a new key pair  
2641 internally and provide the public part. The key pair shall be compatible with the allowed  
2642 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF  
2643 authentication.

2644 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on  
2645 its own, then it is not capable of using certificates. The Device shall advertise this fact both by

2646 setting the 0x8 bit position in the sct Property of "/oic/sec/doxm" to 0, and return an error that  
2647 the "/oic/sec/csr" resource does not exist.

2648 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using  
2649 the same credid, to maintain the association with the private key. The credential type  
2650 ("oic.sec.cred") used to transfer certificates in Figure 29 is also used to transfer role  
2651 certificates, by including multiple credentials in the POST from CMS to Device. Identity  
2652 certificates shall be stored with the credusage Property set to "oic.sec.cred.cert" and role  
2653 certificates shall be stored with the credusage Property set to "oic.sec.cred.rolecert".



2654

2655 **Figure 29 – Client-directed Certificate Transfer**

### 2656 9.4.6 CRL Provisioning

2657 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the  
2658 function to register revocation certificates, to sign CRL and to transfer it to Devices.

2659 The CMS sends the CRL to the Device.

2660 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2661 – change of issuer name
- 2662 – change of association between Devices and CA
- 2663 – certificate compromise
- 2664 – suspected compromise of the corresponding private key

2665 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In  
2666 some special cases, Devices may request CRL to a given CMS.

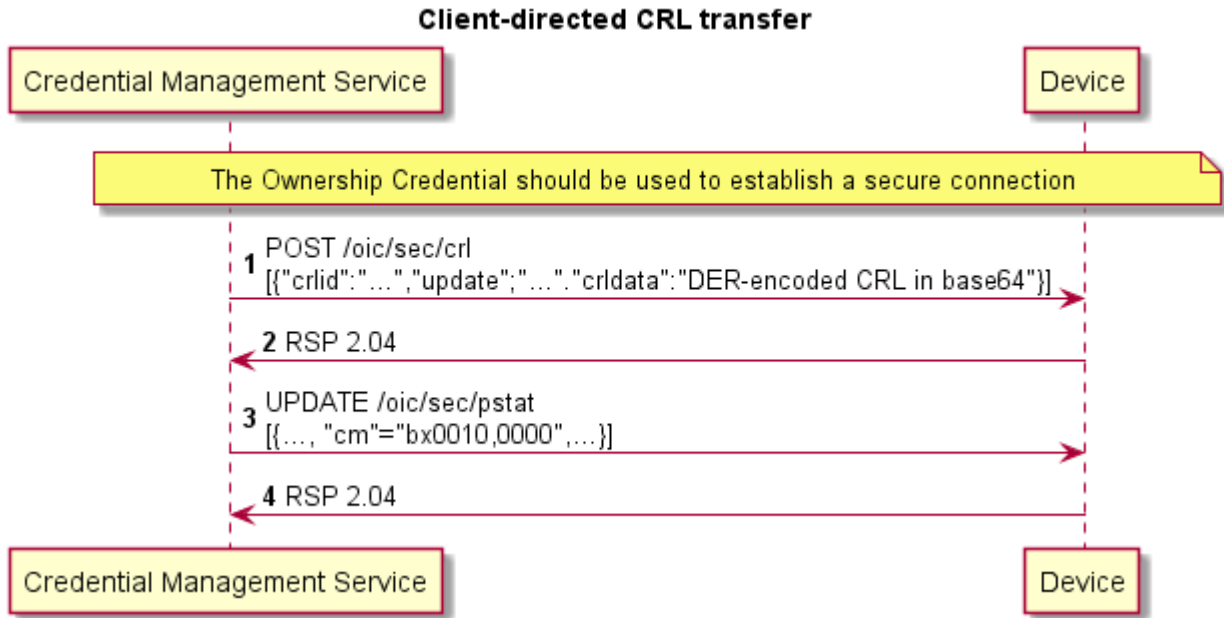
2667 There are two options to update and deliver CRL;

- 2668 – CMS pushes CRL to each Device
- 2669 – each Device periodically requests to update CRL

2670 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

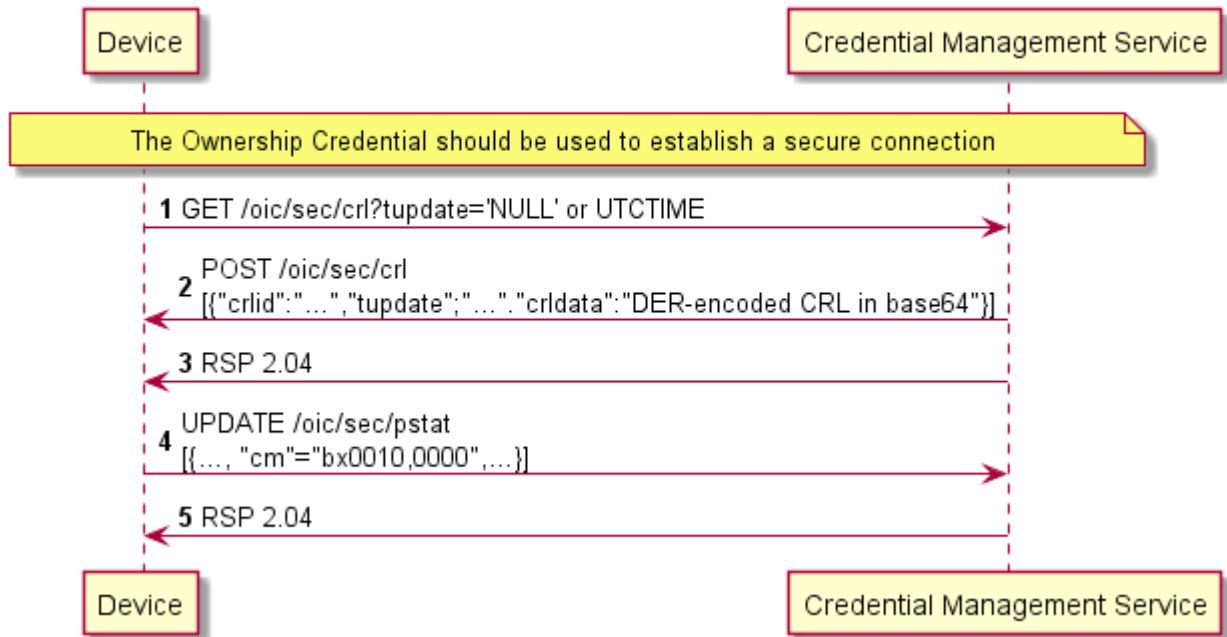
2671 1) The CMS may retrieve the CRL Resource Property.

2672 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.  
 2673  
 2674 ---



2675 **Figure 30 – Client-directed CRL Transfer**  
 2676 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.  
 2677 1) The Device retrieves the CRL Resource Property "tupdate" to the CMS.  
 2678 2) If the CMS recognizes the updated CRL information after the designated "tupdate" time, it  
 2679 may transfer its CRL to the Device.

### Server-directed CRL transfer



2680  
2681  
2682

Figure 31 – Server-directed CRL Transfer



## 2683 **10 Device Authentication**

### 2684 **10.1 Device Authentication General**

2685 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the  
2686 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or  
2687 more roles that the server can use in access control decisions. Roles may be asserted when the  
2688 Device authentication is done with certificates.

### 2689 **10.2 Device Authentication with Symmetric Key Credentials**

2690 When using symmetric keys to authenticate, the Server Device shall include the  
2691 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client  
2692 shall validate that it has a credential with the Subject ID set to the Server's Device ID, and a  
2693 credential type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error  
2694 or other suitable error.

2695 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that  
2696 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a  
2697 credential with the matching Subject ID and type. If it does not, the Server shall respond with an  
2698 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS  
2699 protocol, and both Client and Server shall compute the resulting premaster secret.

### 2700 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2701 When using raw asymmetric keys to authenticate, the Client and the Server shall include a  
2702 suitable public key from a credential that is bound to their Device. Each Device shall verify that  
2703 the provided public key matches the `PublicData` field of a credential they have, and use the  
2704 corresponding Subject ID of the credential to identify the peer Device.

### 2705 **10.4 Device Authentication with Certificates**

#### 2706 **10.4.1 Device Authentication with Certificates General**

2707 When using certificates to authenticate, the Client and Server shall each include their certificate  
2708 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.  
2709 Each Device shall validate the certificate chain presented by the peer Device. Each certificate  
2710 signature shall be verified until a public key is found within the `"/oic/sec/cred"` Resource with the  
2711 `"oic.sec.cred.trustca"` credusage. Credential Resource found in `"/oic/sec/cred"` are used to  
2712 terminate certificate path validation. Also, the validity period and revocation status should be  
2713 checked for all above certificates, but at this time a failure to obtain a certificate's revocation  
2714 status (CRL or OCSP response) MAY continue to allow the use of the certificate if all other  
2715 verification checks succeed.

2716 If available, revocation information should be used to verify the revocation status of the certificate.  
2717 The URL referencing the revocation information should be retrieved from the certificate (via the  
2718 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used  
2719 to gather relevant revocation information like CRLs or OCSP responses.

2720 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2721 Devices must follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2722 particular:

- 2723 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is  
2724 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate  
2725 chain **MUST** be rejected. If the `pathLenConstraint` field is present, Devices will confirm the  
2726 number of certificates between this certificate and the End-Entity certificate is less than or  
2727 equal to `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity

2728 certificate can be issued by this certificate. If the pathLenConstraint field is absent, there is no  
2729 limit to the chain length.

2730 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is  
2731 present, and that the keyCertSign bit is asserted.

2732 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.  
2733 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt  
2734 validation with the public keys of possible issuer certificates whose subject name equals the  
2735 issuer name of this certificate.

2736 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage  
2737 (EKU) suitable to the purpose for which it is being presented. An End-Entity certificate which  
2738 contains no EKU extension is not valid for any purpose and must be rejected. Any certificate  
2739 which contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other  
2740 valid EKUs are also present.

2741 – Devices MUST verify "transitive EKU" for certificate chains. Issuer certificates (any certificate  
2742 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the  
2743 certificate chain is being presented. An issuer certificate is valid for a purpose if it contains an  
2744 EKU extension and the EKU OID for that purpose is listed in the extension, OR it does not  
2745 have an EKU extension. An issuer certificate SHOULD contain an EKU extension and a  
2746 complete list of EKUs for the purposes for which it is authorized to issue certificates. An  
2747 issuer certificate without an EKU extension is valid for all purposes; this differs from End-  
2748 Entity certificates without an EKU extension.

2749 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2750 If the Device does not recognize an extension, it must examine the `critical` field. If the field is  
2751 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the  
2752 certificate as if the extension were absent and proceed accordingly. This applies to all certificates  
2753 in a chain.

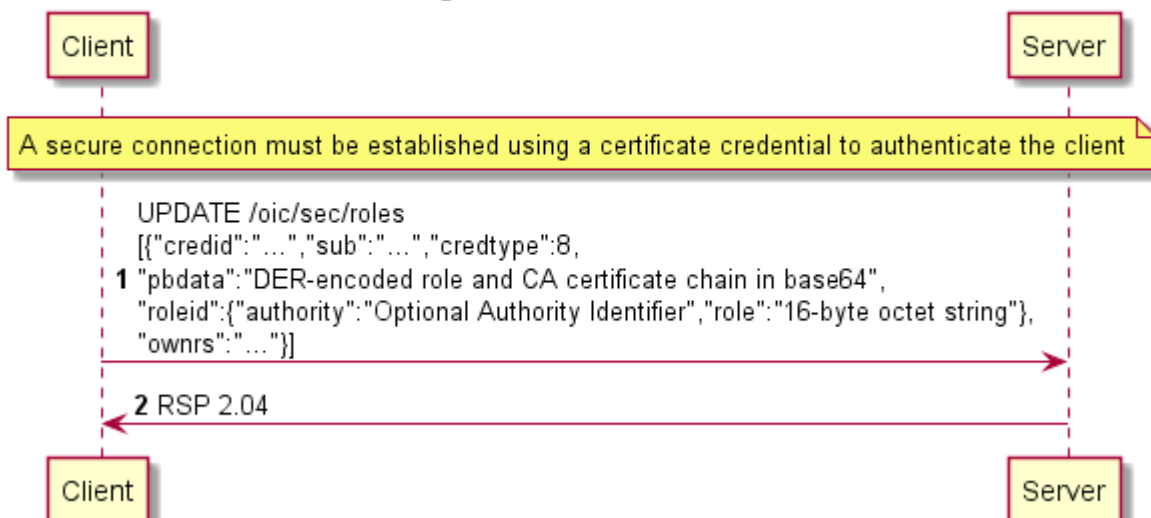
2754 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

#### 2755 **10.4.2 Role Assertion with Certificates**

2756 This clause describes role assertion by a client to a server using a certificate role credential. If a  
2757 server does not support the certificate credential type, clients should not attempt to assert roles  
2758 with certificates.

2759 Following authentication with a certificate, a client may assert one or more roles by updating the  
2760 server's roles resource with the role certificates it wants to use. The role credentials must be  
2761 certificate credentials and shall include a certificate chain. The server shall validate each  
2762 certificate chain as specified in clause 10.3. Additionally, the public key in the End-Entity  
2763 certificate used for Device authentication must be identical to the public key in all role (End-Entity)  
2764 certificates. Also, the subject distinguished name in the End-Entity authentication and role  
2765 certificates must match. The roles asserted are encoded in the subjectAltName extension in the  
2766 certificate. The subjectAltName field can have multiple values, allowing a single certificate to  
2767 encode multiple roles that apply to the client. The server shall also check that the EKU extension  
2768 of the role certificate(s) contains the value 1.3.6.1.4.1.44924.1.7 (see clause 9.4.2.2) indicating  
2769 the certificate may be used to assert roles. Figure 32 describes how a client Device asserts roles  
2770 to a server.

## Asserting Certificate Role Credentials



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

2771  
2772

2773 Additional comments for Figure 32

- 2774 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If  
2775 the server does not support certificate credentials, it should return "501 Not Implemented"
- 2776 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration  
2777 shall not exceed the validity period of the role certificate. When fresh CRL information is  
2778 obtained, the certificates in "/oic/sec/roles" should be checked, and the role removed if the  
2779 certificate is revoked or expired.
- 2780 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role  
2781 by a client. It is recommended that servers use the validity period of the certificate as a  
2782 duration, effectively allowing the CMS to decide the duration.
- 2783 4) The format of the data sent in the create call shall be a list of credentials ("oic.sec.cred", see  
2784 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not  
2785 be present. For fields that are duplicated in the "oic.sec.cred" object and the certificate, the  
2786 value in the certificate shall be used for validation. For example, if the Period field is set in  
2787 the credential, the server shall treat the validity period in the certificate as authoritative.  
2788 Similar for the roleid data (authority, role).
- 2789 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)
- 2790 6) Clients may GET the "/oic/sec/roles" resource to determine the roles that have been  
2791 previously asserted. An array of credential objects shall be returned. If there are no valid  
2792 certificates corresponding to the currently connected and authenticated Client's identity, then  
2793 an empty array (i.e. []) shall be returned.

### 2794 10.4.3 OCF PKI Roots

2795 This clause intentionally left empty.

### 2796 10.4.4 PKI Trust Store

2797 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store  
2798 the OCF Root CA certificates in the oic/sec/cred resource and SHOULD physically store this  
2799 resource in a hardened memory location where the certificates cannot be tampered with.

2800 **10.4.5 Path Validation and extension processing**

2801 Devices SHALL follow the certificate path validation algorithm in clause 6 of IETF RFC 5280. In  
2802 addition, the following are best practices and SHALL be adhered to by any OCF-compliant  
2803 application handling digital certificates

2804 – Validity Period checking

2805 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and  
2806 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,  
2807 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2808 – Revocation checking

2809 Relying applications SHOULD check the revocation status for all certificates, but at this time,  
2810 an application MAY continue to allow the use of the certificate upon a failure to obtain a  
2811 certificate's revocation status (CRL or OCSP response), if all other verification checks  
2812 succeed.

2813 – basicConstraints

2814 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify  
2815 that the basicConstraints extension is present, flagged critical, and that the cA boolean value  
2816 in the extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2817 If the pathLenConstraint field is present, Devices will confirm the number of certificates  
2818 between this certificate and the End-Entity certificate is less than or equal to  
2819 pathLenConstraint. In particular, if pathLenConstraint is zero, only an End-Entity certificate  
2820 can be issued by this certificate. If the pathLenConstraint field is absent, there is no limit to  
2821 the chain length.

2822 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged  
2823 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a  
2824 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a  
2825 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a  
2826 null value.

2827 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-  
2828 compliant Root CA certificates SHALL NOT contain a pathLenConstraint. This allows  
2829 additional tiers of Intermediate CAs to be implemented in the future without changing the Root  
2830 CA trust anchors, should such a requirement emerge.

2831 – keyUsage

2832 For all certificates, Devices shall verify that the key usage extension is present and flagged  
2833 critical.

2834 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits  
2835 SHALL be asserted.

2836 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be  
2837 asserted.

2838 – extendedKeyUsage:

2839 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be  
2840 rejected.

2841 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)  
2842 are required for compatibility with various TLS implementations.

2843 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)  
2844 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be  
2845 present and End-Entity certificates with ECU extensions containing both OIDs SHALL be  
2846 rejected.

2847 – certificatePolicies  
2848 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one  
2849 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)  
2850 corresponding to the version of the OCF Certificate Policy under which it was issued.  
2851 Additional manufacturer-specific CP OIDs may also be populated.

## 2852 **10.5 Device Authentication with OCF Cloud**

### 2853 **10.5.1 Device Authentication with OCF Cloud General**

2854 The mechanisms for Device Authentication in clauses 10.2, 10.3 and 10.4 imply that a Device is  
2855 authorized to communicate with any other Device meeting the criteria provisioned in  
2856 "/oic/sec/cred"; the "/oic/sec/acl2" Resource (or "/oic/sec/acl1" resource of OIC1.1 Servers) are  
2857 additionally used to restrict access to specific Resources. The present clause describes Device  
2858 authentication for OCF Cloud, which uses slightly different criteria as described in clause 5. A  
2859 Device accessing an OCF Cloud shall establish a TLS session. The mutual authenticated TLS  
2860 session is established using Server certificate and Client certificate.

2861 Each Device is identified based on the Access Token it is assigned during Device Registration.  
2862 The OCF Cloud holds an OCF Cloud association table that maps Access Token, User ID and  
2863 Device ID. The Device Registration shall happen while the Device is in RFNOP state. After  
2864 Device Registration, the updated Access Token, Device ID and User ID are used by the Device  
2865 for the subsequent connection with the OCF Cloud.

### 2866 **10.5.2 Device Connection with the OCF Cloud**

2867 The Device should establish the TLS connection using the certificate based credential. The  
2868 connection should be established after Device is provisioned by Mediator.

2869 The TLS session is established between Device and the OCF Cloud as specified in IETF RFC  
2870 8323. The OCF Cloud is expected to provide certificate signed by trust anchor that is present in  
2871 cred entries of the Device. These cred entries are expected to be configured by the Mediator.

2872 The Device shall validate the OCF Cloud's identity based on the credentials that are contained in  
2873 "/oic/sec/cred" Resource entries of the Device.

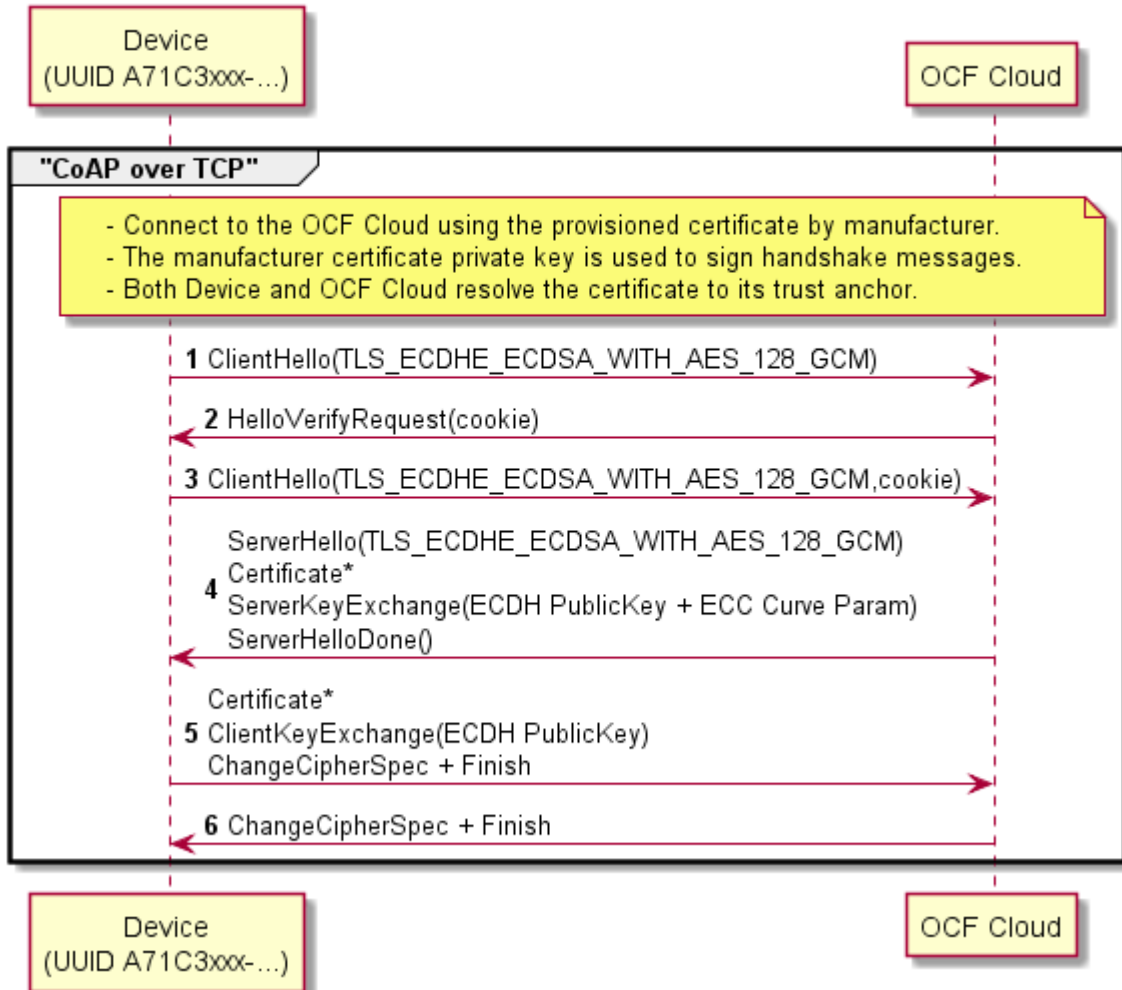
2874 The OCF Cloud is expected to validate the manufacturer certificate provided by the Device.

2875 The assumption is that the OCF Cloud User trusts the OCF Cloud that the Device connects. The  
2876 OCF Cloud connection should not happen without the consent of the OCF Cloud User. The  
2877 assumption is that the OCF Cloud User has either service agreement with the OCF Cloud  
2878 provider or uses manufacturer provided OCF Cloud.

2879 If authentication fails, the "clec" Property of "oic.r.coapcloudconf" Resource on the Device shall  
2880 be updated about the failed state, if it is supported by the Device. If authentication succeeds, the  
2881 Device and OCF Cloud should establish an encrypted link in accordance with the negotiated  
2882 cipher suite.

2883 Figure 33 depicts sequence for Device connection with OCF Cloud and steps described in  
2884 Table 22.

### Device Connection with OCF Cloud



2885

2886

**Figure 33 – Device connection with OCF Cloud**

2887

**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

2888

#### 10.5.3 Security Considerations

2889 When an OCF Server receives a request sent via the OCF Cloud, then the OCF Server permits  
 2890 that request using the identity of the OCF Cloud rather than the identity of the OCF Client. If  
 2891 there is no mechanism through which the OCF Cloud permits only those interactions which the  
 2892 user intends between OCF Clients and OCF Server via the OCF Cloud, and denies all other  
 2893 interactions, then OCF Clients might get elevated privileges by submitting a request via the OCF  
 2894 Cloud. This is highly undesirable from the security perspective. Consequently, OCF Cloud  
 2895 implementations are expected to provide some mechanism through which the OCF Cloud  
 2896 prevents OCF Clients getting elevated privileges when submitting a request via the OCF Cloud.  
 2897 In the present document release, the details of the mechanism are left to the implementation.

2898 The security considerations about the manufacturer certificate as described in 7.3.6.5 are also  
2899 applicable in the Device authentication with the OCF Cloud.

2900 The Device should validate the OCF Cloud's TLS certificate as defined by IETF RFC 6125 and in  
2901 accordance with its requirements for Server identity authentication.

2902 The "uid" and "di" Property Value of "/oic/d" Resource may be considered personally identifiable  
2903 information in some regulatory regions, and the OCF Cloud is expected to provide protections  
2904 appropriate to its governing regulatory bodies.

2905

2906 **11 Message Integrity and Confidentiality**

2907 **11.1 Preamble**

2908 Secured communications between Clients and Servers are protected against eavesdropping,  
2909 tampering, or message replay, using security mechanisms that provide message confidentiality  
2910 and integrity.

2911 **11.2 Session Protection with DTLS**

2912 **11.2.1 DTLS Protection General**

2913 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices  
2914 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See  
2915 11.3 for a list of required and optional cipher suites for message communication.

2916 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1  
2917 or lower.

2918 Multicast session semantics are not yet defined in this version of the security document.

2919 **11.2.2 Unicast Session Semantics**

2920 For unicast messages between a Client and a Server, both Devices shall authenticate each other.  
2921 See clause 10 for details on Device Authentication.

2922 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.  
2923 The sending Device shall encrypt and authenticate messages as defined by the selected cipher  
2924 suite and the receiving Device shall verify and decrypt the messages before processing them.

2925 **11.2.3 Cloud Session Semantics**

2926 The messages between the OCF Cloud and Device shall be exchanged only if the Device and  
2927 OCF Cloud authenticate each other as described in 10.4.3. The asymmetric cipher suites as  
2928 described in 11.3.5 shall be employed for establishing a secured session and for  
2929 encrypting/decrypting between the OCF Cloud and the Device. The OCF Endpoint sending the  
2930 message shall encrypt and authenticate the message using the cipher suite as described in  
2931 11.3.5 and the OCF Endpoint shall verify and decrypt the message before processing it.

2932 **11.3 Cipher Suites**

2933 **11.3.1 Cipher Suites General**

2934 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher  
2935 suites allowed during ownership transfer and normal operation. The following RFCs provide  
2936 additional information about the cipher suites used in OCF.

2937 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2938 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2939 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and  
2940 PSKs

2941 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2942 **11.3.2 Cipher Suites for Device Ownership Transfer**

2943 **11.3.2.1 Just Works Method Cipher Suites**

2944 The Just Works OTM may use the following (D)TLS cipher suites.

2945 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256,



2946 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256

2947 All Devices s supporting Just Works OTM shall implement:

2948 TLS\_ECDH\_ANON\_WITH\_AES\_128\_CBC\_SHA256 (with the value 0xFF00)

2949 All Devices s supporting Just Works OTM should implement:

2950 TLS\_ECDH\_ANON\_WITH\_AES\_256\_CBC\_SHA256 (with the value 0xFF01)

2951 **11.3.2.2 Random PIN Method Cipher Suites**

2952 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2953 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2954 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2955 All Devices s supporting Random Pin Based OTM shall implement:

2956 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256

2957 **11.3.2.3 Certificate Method Cipher Suites**

2958 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

2959 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

2960 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2961 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2962 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2963 Using the following curve:

2964 secp256r1 (See IETF RFC 4492)

2965 All Devices s supporting Manufacturer Certificate Based OTM shall implement:

2966 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

2967 Devices supporting Manufacturer Certificate Based OTM should implement:

2968 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2969 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2970 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2971 **11.3.3 Cipher Suites for Symmetric Keys**

2972 The following cipher suites are defined for (D)TLS communication using PSKs:

2973 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2974 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2975 TLS\_PSK\_WITH\_AES\_128\_CCM\_8, (\* 8 OCTET Authentication tag \*)

2976 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

2977 TLS\_PSK\_WITH\_AES\_128\_CCM, (\* 16 OCTET Authentication tag \*)

2978 TLS\_PSK\_WITH\_AES\_256\_CCM,

2979 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

2980 All Devices shall implement the following:

2981 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2982

2983 Devices should implement the following:

2984 TLS\_ECDHE\_PSK\_WITH\_AES\_128\_CBC\_SHA256,

2985 TLS\_ECDHE\_PSK\_WITH\_AES\_256\_CBC\_SHA256,

2986 TLS\_PSK\_WITH\_AES\_128\_CCM\_8,

2987 TLS\_PSK\_WITH\_AES\_256\_CCM\_8,

2988 TLS\_PSK\_WITH\_AES\_128\_CCM,

2989 TLS\_PSK\_WITH\_AES\_256\_CCM

#### 2990 **11.3.4 Cipher Suites for Asymmetric Credentials**

2991 The following cipher suites are defined for (D)TLS communication with asymmetric keys or  
2992 certificates:

2993 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8,

2994 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

2995 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

2996 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

2997 Using the following curve:

2998 secp256r1 (See IETF RFC 4492)

2999 All Devices supporting Asymmetric Credentials shall implement:

3000 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8

3001 All Devices supporting Asymmetric Credentials should implement:

3002 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM\_8,

3003 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM,

3004 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CCM

#### 3005 **11.3.5 Cipher suites for OCF Cloud Credentials**

3006 The following cipher suites are defined for TLS communication with certificates:

3007 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,

3008 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,

3009 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,

3010 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,

3011 TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,

3012 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

3013 All Devices supporting OCF Cloud Certificate Credentials shall implement:

3014 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256

3015 All Devices supporting OCF Cloud Certificate Credentials should implement:

3016 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256,

3017 TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256,

3018 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384,  
3019 TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384,  
3020 TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  
3021

3022 **12 Access Control**

3023 **12.1 ACL Generation and Management**

3024 This clause will be expanded in a future version of the document.

3025 **12.2 ACL Evaluation and Enforcement**

3026 **12.2.1 ACL Evaluation and Enforcement General**

3027 The Server enforces access control over application Resources before exposing them to the  
3028 requestor. The Security Layer in the Server authenticates the requestor when access is received  
3029 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL  
3030 entries that specify the requestor's identity, role or may match authenticated requestors using a  
3031 subject wildcard.

3032 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a  
3033 subject wildcard match of anonymous requestors.

3034 Access is denied if a requested Resource is not matched by an ACL entry.

3035 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources  
3036 may be granted prior to provisioning of ACL Resources.

3037 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries  
3038 (ACE2) that employ a Resource matching algorithm that uses an array of Resource references to  
3039 match Resources to which the ACE2 access policy applies. Matching consists of comparing the  
3040 values of the ACE2 "resources" Property (see clause 13) to the requested Resource. Resources  
3041 are matched in two ways:

- 3042 1) host reference (href)
- 3043 2) resource wildcard (wc).

3044 **12.2.2 Host Reference Matching**

3045 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for  
3046 Resource matching.

3047 – The href Property shall be used to find an exact match of the Resource name if present.

3048 **12.2.3 Resource Wildcard Matching**

3049 When present, a wildcard (wc) expression shall be used to match multiple Resources using a  
3050 wildcard Property contained in the "oic.sec.ace2.resource-ref" structure.

3051 A wildcard expression may be used to match multiple Resources using a wildcard Property  
3052 contained in the "oic.sec.ace2.resource-ref" structure. The wildcard matching strings are defined  
3053 in Table 23.

3054 **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 all Discoverable Non-Configuration Resources which expose at least one Unsecure OCF Endpoint.
" * "	Shall match all Non-Configuration Resources.

3055 NOTE Discoverable resources appear in the "/oic/res" Resource, while non-discoverable resources may appear in  
3056 other collection resources but do not appear in the /res collection.

#### 3057 **12.2.4 Multiple Criteria Matching**

3058 If the ACE2 "resources" Property contains multiple entries, then a logical OR shall be applied for  
3059 each array element. For example, if a first array element of the "resources" Property contains  
3060 "href="/a/light" and the second array element of the "resources" Property contains "href="/a/led",  
3061 then Resources that match either of the two "href" criteria shall be included in the set of matched  
3062 Resources.

3063 Example 1 JSON for Resource matching

```
3064 {  
3065 //Matches Resources named "/x/door1" or "/x/door2"  
3066   "resources":[  
3067     {  
3068       "href":"/x/door1"  
3069     },  
3070     {  
3071       "href":"/x/door2"  
3072     },  
3073   ]  
3074 }
```

3075 Example 2 JSON for Resource matching

```
3076 {  
3077   // Matches all Resources  
3078   "resources":[  
3079     {  
3080       "wc":"**"  
3081     }  
3082   ]  
3083 }
```

#### 3084 **12.2.5 Subject Matching using Wildcards**

3085 When the ACE subject is specified as the wildcard string "\*" any requestor is matched. The OCF  
3086 server may authenticate the OCF client, but is not required to.

3087 Examples: JSON for subject wildcard matching

```
3088 //matches all subjects that have authenticated and confidentiality protections in place.  
3089 "subject" : {  
3090   "conntype" : "auth-crypt"  
3091 }  
3092 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
3093 "subject" : {  
3094   "conntype" : "anon-clear"  
3095 }
```

#### 3096 **12.2.6 Subject Matching using Roles**

3097 When the ACE subject is specified as a role, a requestor shall be matched if either:

3098 1) The requestor authenticated with a symmetric key credential, and the role is present in the  
3099 roleid Property of the credential's entry in the credential resource, or

3100 2) The requestor authenticated with a certificate, and a valid role certificate is present in the  
3101 roles resource with the requestor's certificate's public key at the time of evaluation. Validating  
3102 role certificates is defined in 10.3.1.

## 3103 **12.2.7 ACL Evaluation**

### 3104 **12.2.7.1 ACE2 matching algorithm**

3105 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 3106 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these  
3107 ACE2 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the  
3108 signature, at least once, following update of the "/oic/sec/sacl" Resource.
- 3109 2) The local "/oic/sec/acl2" Resource contributes its ACE2 entries for matching.
- 3110 3) Access shall be granted when all these criteria are met:
  - 3111 a) The requestor is matched by the ACE2 "subject" Property.
  - 3112 b) The requested Resource is matched by the ACE2 resources Property and the requested  
3113 Resource shall exist on the local Server.
  - 3114 c) The "period" Property constraint shall be satisfied.
  - 3115 d) The "permission" Property constraint shall be applied.

3116 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching  
3117 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then  
3118 UNION (Perm1, Perm2)=CRUDN.

3119 The Server shall enforce access based on the effective permissions granted.

3120 Batch requests to Resource containing Links require additional considerations when accessing  
3121 the linked Resources. ACL considerations for batch request to the Atomic Measurement  
3122 Resource Type are provided in clause 12.2.7.2. ACL considerations for batch request to the  
3123 Collection Resource Type are provided in 12.2.7.3.

### 3124 **12.2.7.2 ACL considerations for batch request to the Atomic Measurement Resource 3125 Type**

3126 The present clause shall apply to any Resource Type based on the Atomic Measurement  
3127 Resource Type.

3128 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only  
3129 local references and there is an ACE matching the Atomic Measurement Resource which permits  
3130 the request, then the corresponding requests to the linked Resources of the Atomic Measurement  
3131 Resource shall be permitted by the OCF Server. That is, the request to each linked Resource is  
3132 permitted regardless of whether there is an ACE configured on the OCF Server which would  
3133 permit a corresponding request from the OCF Client (which sent the batch request to the Atomic  
3134 Measurement Resource) addressing the linked Resource.

### 3135 **12.2.7.3 ACL considerations for batch request to the Collection Resource Type**

3136 The present clause shall apply to any Resource Type based on the Collection Resource Type.

3137 If an OCF Server receives a batch request to a Collection Resource containing only local  
3138 references and there is an ACE matching the Collection Resource which permits the request,  
3139 then the corresponding requests to the linked Resources of the Collection Resource shall be  
3140 permitted by the OCF Server. That is, the request to each linked Resource is permitted  
3141 regardless of whether there is an ACE configured on the OCF Server which would permit a  
3142 corresponding request from the OCF Client (which sent the batch request to the Collection  
3143 Resource) addressing the linked Resource.



3145 **13 Security Resources**

3146 **13.1 Security Resources General**

3147 OCF Security Resources are shown in Figure 34.

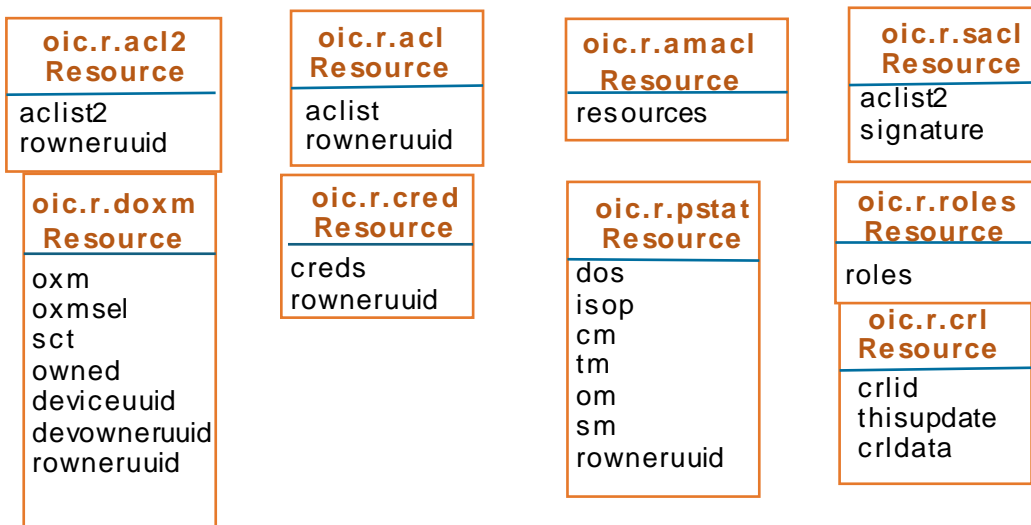
3148 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

3149 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

3150 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

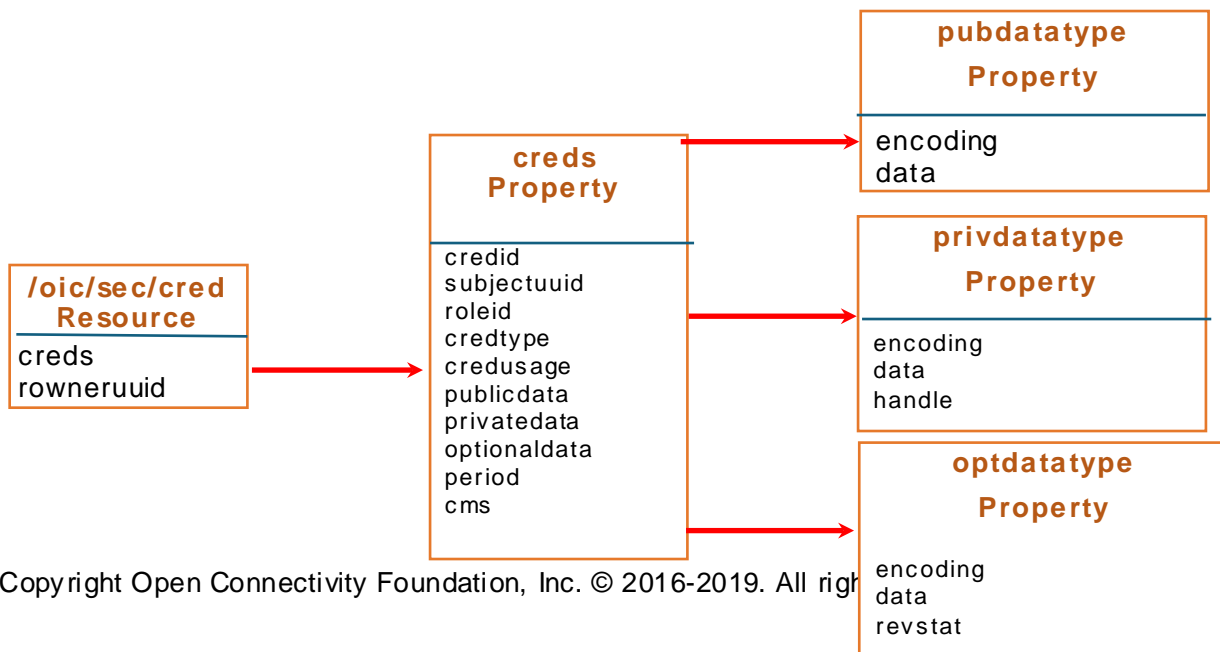
3151 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

3152



3153

**Figure 34 – OCF Security Resources**

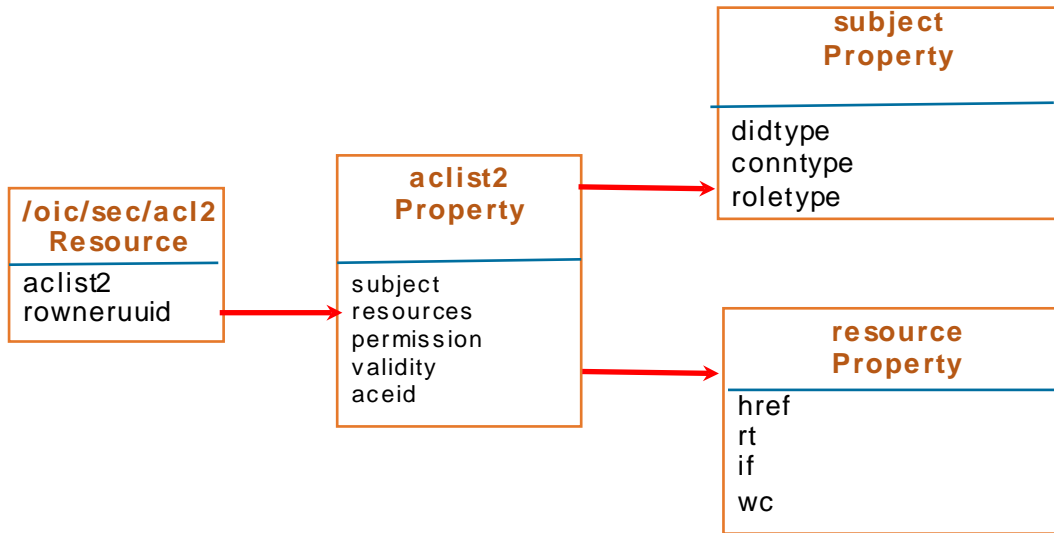




3154

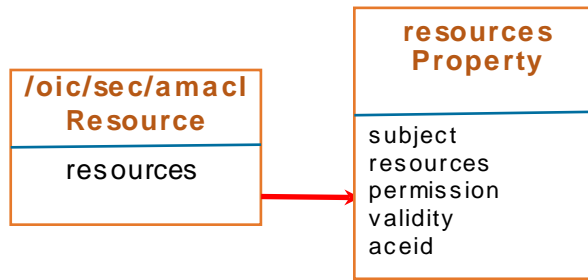
**Figure 35 – "/oic/sec/cred" Resource and Properties**

3155



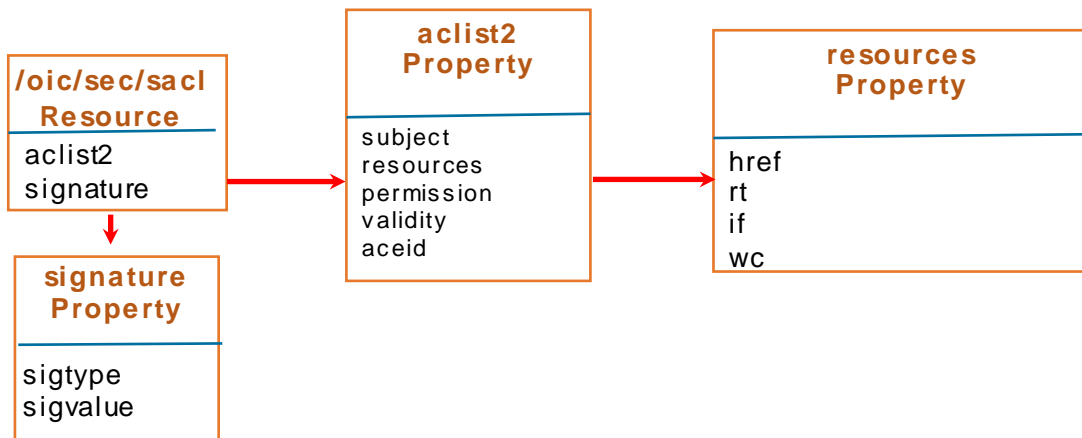
3156

**Figure 36 – "/oic/sec/acl2" Resource and Properties**



3157

**Figure 37 – "/oic/sec/amacl" Resource and Properties**



3158

**Figure 38 – "/oic/sec/sacl" Resource and Properties**

3159 **13.2 Device Owner Transfer Resource**

3160 **13.2.1 Device Owner Transfer Resource General**

3161 The "/oic/sec/doxm" Resource contains the set of supported Device OTMs.

3162 Resource discovery processing respects the CRUDN constraints supplied as part of the security  
3163 Resource definitions contained in this document.

3164 "/oic/sec/doxm" Resource is defined in Table 24.

3165

**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

3166 Table 25 defines the Properties of the "/oic/sec/doxm" Resource.

3167

**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 its 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 DOTS fails the Server shall transition device state to RESET.
					RFPRO	R	n/a
					RFNOP	R	n/a
					SRESET	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.
Device Ownership Status	owned	Boolean	T F	Yes	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
					RFNOP	R	n/a
					SRESET	R	n/a
Device UUID	deviceuuid	String	oic.sec.didtype	Yes	RESET	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
					SRESET	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.

3168 Table 26 defines the Properties of the "/oic/sec/didtype".

3169 **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

3170 The oxms Property contains a list of OTM where the entries appear in the order of preference.  
3171 This Property contains the higher priority methods appearing before the lower priority methods.  
3172 The DOTS queries this list at the time of onboarding and selects the most appropriate method.

3173 The DOTS shall update the oxmsel Property of the "/oic/sec/doxm" Resource with the OTM that  
3174 was used to onboard the Device.

3175 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3176 <DoxmType> ::= <NSS>
3177 <NSS> ::= <Identifier> | { {<NID>"."} <NameSpaceQualifier> "."} <Method>
3178 <NID> ::= <Vendor-or-Organization>
3179 <Identifier> ::= INTEGER
3180 <NameSpaceQualifier> ::= String
3181 <Method> ::= String
3182 <Vendor-Organization> ::= String

```

3183 When an OTM successfully completes, the *owned* Property is set to "1" (TRUE). Consequently,  
3184 subsequent attempts to take ownership of the Device will fail.

3185 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in  
3186 the "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource  
3187 is UPDATED to non-nil UUID value.

3188 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource  
3189 after it has updated the devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-  
3190 nil-UUID value.

3191 The Device vendor shall determine that the Device identifier (deviceuuid) is persistent (not  
3192 updatable) or that it is non-persistent (updatable by the owner transfer service – aka. DOTS).

3193 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall  
3194 fail with the error PROPERTY\_NOT\_FOUND.

3195 If the deviceuuid Property of the "/oic/sec/doxm" Resource is non-persistent, the request to  
3196 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is  
3197 RESET. If the UPDATE to deviceuuid Property of the "/oic/sec/doxm" Resource fails while in the  
3198 RFOTM Device state the device state shall transition to RESET where the Server shall set the  
3199 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g.  
3200 "00000000-0000-0000-0000-000000000000").

3201 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the  
3202 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the deviceuuid  
3203 Property of the "/oic/sec/doxm" Resource each time the device enters RESET Device state. The  
3204 temporary deviceuuid value is used while the device state is in the RESET state and while in the  
3205 RFOTM device state until the DOTS establishes a secure OTM connection. The DOTS should  
3206 RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm" Resource after it has  
3207 updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-nil-UUID value.

3208 The deviceuuid Property of the "/oic/sec/doxm" Resource shall expose a persistent value(i.e. is  
3209 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the DOTS via  
3210 an OCF Interface to the deviceuuid Property of the "/oic/sec/doxm" Resource during RFOTM  
3211 Device state.).

3212 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a  
3213 secure OTM connection and UPDATES the devowneruuid Property to a non-nil UUID value.  
3214 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the deviceuuid Property of  
3215 the "/oic/sec/doxm" Resource shall reveal the persistent or semi-persistent value to authenticated  
3216 requestors and shall reveal the temporary non-repeated value to unauthenticated requestors.

3217 See 13.16 for additional details related to privacy sensitive considerations.

### 3218 **13.2.2 Persistent and Semi-Persistent Device Identifiers**

3219 The Device vendor determines whether a device identifier can be set by a configuration tool or  
3220 whether it is immutable. If it is an immutable value this document refers to it as a persistent  
3221 device identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four  
3222 device identifiers that could be considered persistent or semi-persistent:

- 3223 1) "deviceuuid" Property of "/oic/sec/doxm"
- 3224 2) "di" Property of "/oic/d"
- 3225 3) "piid" Property of "/oic/d"
- 3226 4) "pi" Property of "/oic/p"

### 3227 **13.2.3 Onboarding Considerations for Device Identifier**

3228 The deviceuuid is used to onboard the Device. The other identifiers (di, piid and pi) are not  
3229 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the  
3230 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security  
3231 Domain owner may have a preference for persistent or semi-persistent device identifiers.  
3232 Detecting whether the Device is using persistent or semi-persistent deviceuuid can be achieved  
3233 by attempting to update it.

3234 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is persistent, then an UPDATE  
 3235 request, at the appropriate time during onboarding shall fail with an appropriate error response.

3236 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device  
 3237 state is RFOTM and when devowneruid Property value of the "/oic/sec/doxm" Resource has a  
 3238 non-nil UUID value.

3239 If the "deviceuuid" Property of the "/oic/sec/doxm" Resource is semi-persistent, subsequent to a  
 3240 successful UPDATE request to change it; the Device shall remember the semi-persistent value  
 3241 until the next successful UPDATE request or until the Device state transitions to RESET.

3242 See 13.16 for addition behaviour regarding "deviceuuid".

3243

3244 **13.2.4 OCF defined OTMs**

3245 Table 27 defines the Properties of the "oic.sec.doxmtype".

3246 **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 DOTS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTS 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.oxm.self	4	The manufacturer shall set the /doxm.oxmself 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.

3247 **13.3 Credential Resource**

3248 **13.3.1 Credential Resource General**

3249 The "/oic/sec/cred" Resource maintains credentials used to authenticate the Server to Clients and  
3250 support services as well as credentials used to verify Clients and support services.

3251 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared  
3252 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to  
3253 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3254 In order to provide an interface which allows management of the "creds" Array Property, the  
3255 RETRIEVE, UPDATE and DELETE operations on the "oic.r.cred" Resource shall behave as  
3256 follows:

- 3257 1) A RETRIEVE shall return the full Resource representation, except that any write-only  
3258 Properties shall be omitted (e.g. private key data).
- 3259 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3260 UPDATE request, as follows:
  - 3261 a) If an UPDATE representation includes the "creds" array Property, then:
    - 3262 i) Supplied creds with a "credid" that matches an existing "credid" shall replace  
3263 completely the corresponding cred in the existing "creds" array.
    - 3264 ii) Supplied creds without a "credid" shall be appended to the existing "creds" array, and  
3265 a unique (to the cred Resource) "credid" shall be created and assigned to the new  
3266 cred by the Server. The "credid" of a deleted cred should not be reused, to improve  
3267 the determinism of the interface and reduce opportunity for race conditions.
    - 3268 iii) Supplied creds with a "credid" that does not match an existing "credid" shall be  
3269 appended to the existing "creds" array, using the supplied "credid".
    - 3270 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device  
3271 States in which an UPDATE of the "creds" array Property is always rejected. If OCF  
3272 Device is in a Device State where the Access Mode in this row contains "R", then the  
3273 OCF Device shall reject all UPDATES of the "creds" array Property.
  - 3274 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not  
3275 remove the "oic.r.cred" Resource.
  - 3276 4) A DELETE with one or more "credid" query parameters shall remove the cred(s) with the  
3277 corresponding credid(s) from the "creds" array.
  - 3278 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in  
3279 which a DELETE is always rejected. If OCF Device is in a Device State where the Access  
3280 Mode in this row contains "R", then the OCF Device shall reject all DELETES.

3281 NOTE The "oic.r.cred" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
3282 in ISO/IEC 30118-1:2018.

3283 "/oic.r.cred" Resource is defined in Table 28.

3284 **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

3285 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.

3287 All secure Device accesses shall have a "/oic/sec/cred" Resource that protects the end-to-end  
3288 interaction.

3289 The "/oic/sec/cred" Resource shall be updateable by the service named in its rowneruuid  
3290 Property.

3291 ACLs naming "/oic/sec/cred" Resource should further restrict access beyond CRUDN access  
3292 modes.

3293 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".

3295 Table 31 defines the Properties of "oic.sec.credusagetype".

3296 **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

3297 Table 32 defines the Properties of "oic.sec.pubdatatype".

3298 **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

3299 Table 33 defines the Properties of "oic.sec.privdatatype".

3300 **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

3301 Table 34 defines the Properties of "oic.sec.optdatatype".

3302 **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

3303 Table 35 defines the Properties of "oic.sec.roletype".

3304 **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.

3305 **13.3.2 Properties of the Credential Resource**

3306 **13.3.2.1 Credential ID**

3307 Credential ID (credid) is a local reference to an entry in a creds Property array of the  
3308 "/oic/sec/cred" Resource. The SRM generates it. The credid Property shall be used to  
3309 disambiguate array elements of the creds Property.

3310 **13.3.2.2 Subject UUID**

3311 The subjectuuid Property identifies the Device to which an entry in a creds Property array of the  
3312 "/oic/sec/cred" Resource shall be used to establish a secure session, verify an authentication  
3313 challenge-response or to authenticate an authentication challenge.

3314 A subjectuuid Property that matches the Server's own deviceuuid Property, distinguishes the  
3315 array entries in the creds Property that pertain to this Device.

3316 The subjectuuid Property shall be used to identify a group to which a group key is used to protect  
3317 shared data.

3318 When certificate chain is used during secure connection establishment, the "subjectuuid"  
3319 Property shall also be used to verify the identity of the responder. The presented certificate chain  
3320 shall be accepted, if there is a matching Credential entry on the Device that satisfies all of the  
3321 following:

- 3322 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3323 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity  
3324 certificate in the presented chain. If Subject UUID of the entry is set as a wildcard "\*", this  
3325 condition is automatically satisfied.
- 3326 – Credential Usage of the entry is "oic.sec.cred.trustca".

### 3327 **13.3.2.3 Role ID**

3328 The roleid Property identifies a role that has been granted to the credential.

### 3329 **13.3.2.4 Credential Type**

3330 The credtype Property is used to interpret several of the other Property values whose contents  
3331 can differ depending on credential type. These Properties include publicdata, privatedata and  
3332 optionaldata. The credtype Property value of "0" ("no security mode") is reserved for testing and  
3333 debugging circumstances. Production deployments shall not allow provisioning of credentials of  
3334 type "0". The SRM should introduce checking code that prevents its use in production  
3335 deployments.

### 3336 **13.3.2.5 Public Data**

3337 The publicdata Property contains information that provides additional context surrounding the  
3338 issuance of the credential. For example, it might contain information included in a certificate or  
3339 response data from a CMS. It might contain wrapped data.

### 3340 **13.3.2.6 Private Data**

3341 The privatedata Property contains secret information that is used to authenticate a Device,  
3342 protect data or verify an authentication challenge-response.

3343 The privatedata Property shall not be disclosed outside of the SRM's trusted computing perimeter.  
3344 A secure element (SE) or trusted execution environment (TEE) should be used to implement the  
3345 SRM's trusted computing perimeter. The privatedata contents may be referenced using a handle;  
3346 for example, if used with a secure storage sub-system.

### 3347 **13.3.2.7 Optional Data**

3348 The optionaldata Property contains information that is optionally supplied, but facilitates key  
3349 management, scalability or performance optimization.

### 3350 **13.3.2.8 Period**

3351 The period Property identifies the validity period for the credential. If no validity period is  
3352 specified, the credential lifetime is undetermined. Constrained devices that do not implement a  
3353 date-time capability shall obtain current date-time information from its CMS.

### 3354 **13.3.2.9 Credential Refresh Method Type Definition**

3355 The CMS shall implement the credential refresh methods specified in the crms Property of the  
3356 "oic.sec.creds" array in the "/oic/sec/cred" Resource.

3357 Table 36 defines the values of "oic.sec.crmtpe".

**Table 36 – Value Definition of the "oic.sec.crmtype" 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.rownruuid" 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.rownruuid" 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.rownruuid" 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.rownruuid" Resource to identify the key management service that supports this credential refresh method. The Server uses its "/oic/sec/cred.rownruuid" 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.rownruuid" Resource to identify the key management service that supports this credential refresh method. The Server uses its "/oic/sec/cred.rownruuid" Resource to identify a key management service that supports this credential refresh method.

### 3359 13.3.2.10 Credential Usage

3360 Credential Usage indicates to the Device the circumstances in which a credential should be used.  
3361 Five values are defined:

- 3362 – oic.sec.cred.trustca: This certificate is a trust anchor for the purposes of certificate chain  
3363 validation, as defined in 10.3.
- 3364 – oic.sec.cred.cert: This credusage is used for certificates for which the Device possesses the  
3365 private key and uses it for identity authentication in a secure session, as defined in clause  
3366 10.4.
- 3367 – oic.sec.cred.rolecet: This credusage is used for certificates for which the Device possesses  
3368 the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3369 – oic.sec.cred.mfgtrustca: This certificate is a trust anchor for the purposes of the Manufacturer  
3370 Certificate Based OTM as defined in clause 7.3.6.

3371 – oic.sec.cred.mfgcert: This certificate is used for certificates for which the Device possesses  
3372 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as  
3373 defined in clause 7.3.6.

### 3374 **13.3.3 Key Formatting**

#### 3375 **13.3.3.1 Symmetric Key Formatting**

3376 Symmetric keys shall have the format described in Table 37 and Table 38.

3377 **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.

3378

3379 **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.

#### 3380 **13.3.3.2 Asymmetric Keys**

3381 Asymmetric key formatting is not available in this revision of the document.

#### 3382 **13.3.3.3 Asymmetric Keys with Certificate**

3383 Key formatting is defined by certificate definition.

#### 3384 **13.3.3.4 Passwords**

3385 Password formatting is not available in this revision of the document.

### 3386 **13.3.4 Credential Refresh Method Details**

#### 3387 **13.3.4.1 Provisioning Service**

3388 The resource owner identifies the provisioning service. If the Server determines a credential  
3389 requires refresh and the other methods do not apply or fail, the Server will request re-  
3390 provisioning of the credential before expiration. If the credential is allowed to expire, the Server  
3391 should delete the Resource.

#### 3392 **13.3.4.2 Pre-Shared Key**

##### 3393 **13.3.4.2.1 Pre-Shared Key General**

3394 Using this mode, the current PSK is used to establish a Diffie-Hellman session key in DTLS. The  
3395 TLS\_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3396  $PSK = TLS\_PRF(\text{MasterSecret}, \text{Message}, \text{length});$

3397 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of  
3398 the above ciphersuites.

3399 – Message is the concatenation of the following values:

3400 – RM - Refresh method – I.e. "oic.sec.crm.psk"

3401 – Device ID\_A is the string representation of the Device ID that supplied the DTLS  
3402 ClientHello.

3403 – Device ID\_B is the Device responding to the DTLS ClientHello message

3404 – Length of Message in bytes.

3405 Both Server and Client use the PSK to update the "/oic/sec/cred" Resource's privatedata Property.  
3406 If Server initiated the credential refresh, it selects the new validity period. The Server sends the  
3407 chosen validity period to the Client over the newly established DTLS session so it can update the  
3408 corresponding credential Resource for the Server.

#### 3409 **13.3.4.2.2 Random PIN**

3410 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.  
3411 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session  
3412 key should be used to switch from PIN to PSK mode.

3413 The PIN is randomly generated by the Server and communicated to the Client through an out-of-  
3414 band method. The OOB method used is out-of-scope.

3415 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used  
3416 to generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a  
3417 DTLS ciphersuite that accepts a PSK.

3418  $PPSK = PBKDF2(PRF, PIN, RM, Device\ ID, c, dkLen)$

3419 The PBKDF2 function has the following parameters:

3420 – PRF – Uses the DTLS PRF.

3421 – PIN – Shared between Devices.

3422 – RM - Refresh method – I.e. "oic.sec.crm.rdp"

3423 – Device ID – UUID of the new Device.

3424 – c – Iteration count initialized to 1000, incremented upon each use.

3425 – dkLen – Desired length of the derived PSK in octets.

3426 Both Server and Client use the PPSK to update the "/oic/sec/cred" Resource's PrivateData  
3427 Property. If Server initiated the credential refresh, it selects the new validity period. The Server  
3428 sends the chosen validity period to the Client over the newly established DTLS session so it can  
3429 update its corresponding credential Resource for the Server.

#### 3430 **13.3.4.2.3 SKDC**

3431 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruid  
3432 Property value that matches a CMS that implements SKDC functionality and where the Client  
3433 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request  
3434 message is delivered to the CMS and in response returns the ticket request. The Server updates  
3435 or instantiates an "/oic/sec/cred" Resource guided by the ticket response contents.

#### 3436 **13.3.4.2.4 PKCS10**

3437 A DTLS session is opened to the Server where the "/oic/sec/cred" Resource has an rowneruid  
3438 Property value that matches a CMS that supports the oic.sec.crm.pk10 credential refresh method.  
3439 A PKCS10 formatted message is delivered to the service. After the refreshed certificate is issued,  
3440 the CMS pushes the certificate to the Server. The Server updates or instantiates an  
3441 "/oic/sec/cred" Resource guided by the certificate contents.

3442 **13.3.4.3 Resource Owner**

3443 The Resource Owner Property allows credential provisioning to occur soon after Device  
 3444 onboarding before access to support services has been established. It identifies the entity  
 3445 authorized to manage the "/oic/sec/cred" Resource in response to Device recovery situations.

3446 **13.4 Certificate Revocation List**

3447 **13.4.1 CRL Resource Definition**

3448 Device certificates and private keys are kept in `cred` Resource. CRL is maintained and updated  
 3449 with a separate `crl` Resource that is newly defined for maintaining the revocation list.

3450 "oic.r.crl" Resource is defined in Table 39.

3451 **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

3452 Table 40 defines the Properties of "oic.r.crl".

3453 **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

3454 **13.5 ACL Resources**

3455 **13.5.1 ACL Resources General**

3456 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be  
 3457 expressed using three ACL Resource Types: "/oic/sec/acl2", "/oic/sec/amacl" and "/oic/sec/sacl".  
 3458 The subject (e.g. deviceuuid of the Client) requesting access to a Resource shall be  
 3459 authenticated prior to applying the ACL check. Resources that are available to multiple Clients  
 3460 can be matched using a wildcard subject. All Resources accessible via the unsecured  
 3461 communication endpoint shall be matched using a wildcard subject.

3462 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3463 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3464 **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>

3465 The <DeviceId> token means the requestor must possess a credential that uses <UUID> as its  
3466 identity in order to match the requestor to the <ACE> policy.

3467 The <RoleId> token means the requestor must possess a role credential with <Character> as its  
3468 role in order to match the requestor to the <ACE> policy.

3469 The <Wildcard> token "\*" means any requestor is matched to the <ACE> policy, with or without  
3470 authentication.

3471 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the  
3472 <ACE> policy to Resources.

3473 The <OIC\_LINK> token contains values used to query existence of hosted Resources.

3474 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>  
3475 and <ResourceRef> matching does not produce the empty set match.

3476 Permissions are defined in terms of CREATE ("C"), RETRIEVE ("R"), UPDATE ("U"), DELETE  
3477 ("D"), NOTIFY ("N") and NIL ("-"). NIL is substituted for a permissions character that signifies the  
3478 respective permission is not granted.

3479 The empty set match result defaults to a condition where no access rights are granted.

3480 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.  
3481 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively  
3482 be granted and rescinded according to the pattern.

### 3483 13.5.3 ACL Resource

3484 There are two types of ACLs, "acl" is a list of type "ace" and "acl2" is a list of type "ace2". A  
3485 Device shall not host the /acl Resource.

3486 NOTE The /acl Resource is defined for backward compatibility and use by Provisioning Tools, etc.

3487 In order to provide an interface which allows management of array elements of the "aclist2"  
3488 Property associated with an "/oic/sec/acl2" Resource. The RETRIEVE, UPDATE and DELETE  
3489 operations on the "/oic/sec/acl2" Resource SHALL behave as follows:

- 3490 1) A RETRIEVE shall return the full Resource representation.
- 3491 2) An UPDATE shall replace or add to the Properties included in the representation sent with the  
3492 UPDATE request, as follows:
  - 3493 a) If an UPDATE representation includes the array Property, then:

- 3494 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace  
3495 completely the corresponding ACE in the existing "aces2" array.
- 3496 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array,  
3497 and a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new  
3498 ACE by the Server. The "aceid" of a deleted ACE should not be reused, to improve  
3499 the determinism of the interface and reduce opportunity for race conditions.
- 3500 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be  
3501 appended to the existing "aces2" array, using the supplied "aceid".

3502 The rows in Table 47 defines the Properties of "oic.sec.acl2".

- 3503 iv) Table 47 corresponding to the "aclist2" array Property dictate the Device States in  
3504 which an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is  
3505 in a Device State where the Access Mode in this row contains "R", then the OCF  
3506 Device shall reject all UPDATES of the "aclist2" array Property.

3507 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not  
3508 remove the "oic.r.ace2" Resource.

3509 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the  
3510 corresponding aceid(s) from the "aces2" array.

3511 The rows in Table 47 defines the Properties of "oic.sec.acl2".

3512 5) Table 47 corresponding to the "aclist2" array Property dictate the Device States in which a  
3513 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this  
3514 row contains "R", then the OCF Device shall reject all DELETES.

3515 NOTE The "oic.r.acl2" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces  
3516 defined in ISO/IEC 30118-1:2018.

3517 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no  
3518 entry can be found for the requested Resource for the requestor – e.g. "/oic/sec/acl",  
3519 "/oic/sec/sacl" and "/oic/sec/amacl" do not match the subject and the requested Resource.

3520 It is possible the AMS has an ACL policy that satisfies a resource access request, but the  
3521 necessary ACE has not been provisioned to Server. The Server may open a secure connection to  
3522 the AMS to request ACL provisioning. The Server may use filter criteria that returns a subset of  
3523 the AMS ACL policy. The AMS shall obtain the Server Device ID using the secure connection  
3524 context.

3525 The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS  
3526 to process an "/oic/sec/amacl" Resource. The AMS shall match the AMACL policy and return the  
3527 Permission Property or an error if no match is found.

3528 If the requested Resource is still not matched, the Server returns an error. The requester should  
3529 query the Server to discover the configured AMS services. The Client should contact the AMS to  
3530 request a sacl ("/oic/sec/sacl") Resource. Performing the following operations implement this type  
3531 of request:

3532 1) Client: Open secure connection to AMS.

3533 2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3534 3) AMS: constructs a "/oic/sec/sacl" Resource that is signed by the AMS and returns it in  
3535 response to the RETRIEVE command.

3536 4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3537 5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3538 6) Client: retries original Resource access request. This time the new ACL is included in the  
 3539 local ACL evaluation.

3540 The ACL contained in the "/oic/sec/sacl" Resource should grant longer term access that satisfies  
 3541 repeated Resource requests.

3542 "oic.r.acl" Resource is defined in Table 42.

3543 **Table 42 – Definition of the "oic.r.acl" Resource**

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

3544 Table 43 defines the Properties of "oic.r.acl".

3545 **Table 43 – Properties of the "oic.r.acl" Resource**

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 rowneruid 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 devowneruid 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.
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.
					R	RFPRO	n/a

					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via /doxm devowneruuid Property or the /doxm rowneruuid 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.

3546 Table 44 defines the Properties of "oic.r.ace".

3547 **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.

3548 Table 45 defines the values of "oic.sec.crudntype".

3549 **Table 45 – 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

3550 "oic.sec.acl2" Resource is defined in Table 28.

3551

**Table 46 – 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

3552 Table 47 defines the Properties of "oic.sec.acl2".

3553

**Table 47 – 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 rowneruid 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 devowneruid 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.
rowneruid	uuid	Yes	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
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
			RFOTM	RW	The DOTS should configure the rowneruid 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 devowneruid Property or rowneruid 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 Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3554

3555 Table 48 defines the Properties of "oic.sec.ace2".

3556 **Table 48 – "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.bitmask	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.

3557 Table 49 defines the Properties of "oic.sec.ace2.resource-ref".

3558 **Table 49 – "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.

3559 Table 50 defines the values of "oic.sec.ace2.resource-ref".

3560 **Table 50 – 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

3561 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack  
3562 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's  
3563 request using policies contained in ACL resources.

3564 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified  
3565 Resource references include the device identifier in the href Property that identifies the remote  
3566 Resource Server that hosts the Resource. Partially qualified references mean that the local  
3567 Resource Server hosts the Resource. If a fully qualified resource reference is given, the  
3568 Intermediary enforcing access shall have a secure channel to the Resource Server and the

3569 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource  
3570 access enforcement point.

3571 Resource Servers should include references to Device and ACL Resources where access  
3572 enforcement is to be applied. However, access enforcement logic shall not depend on these  
3573 references for access control processing as access to Server Resources will have already been  
3574 granted.

3575 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and  
3576 modify this Resource. This prevents non-terminating dependency on some other ACL Resource.  
3577 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL  
3578 Resource.

3579 An ACE or ACE2 entry is called currently valid if the validity period of the ACE or ACE2 entry  
3580 includes the time of the request. The validity period in the ACE or ACE2 may be a recurring time  
3581 period (e.g., daily from 1:00-2:00). Matching the resource(s) specified in a request to the  
3582 resource Property of the ACE or ACE2 is defined in clause 12.2. For example, one way they can  
3583 match is if the Resource URI in the request exactly matches one of the resource references in the  
3584 ACE or ACE2 entries.

3585 A request will match an ACE if any of the following are true:

3586 1) The "deviceuuid" Property associated with the secure session matches the "subjectuuid" of  
3587 the ACE; AND the Resource of the request matches one of the "resources" Property of the  
3588 ACE; AND the ACE is currently valid.

3589 2) The ACE "subjectuuid" Property contains the wildcard "\*" character; AND the Resource of the  
3590 request matches one of the "resources" Property of the ACE; AND the ACE is currently valid.

3591 3) When authentication uses a symmetric key credential;

3592 AND the CoAP payload query string of the request specifies a role, which is associated with  
3593 the symmetric key credential of the current secure session;

3594 AND the CoAP payload query string of the request specifies a role, which is contained in the  
3595 "oic.r.cred.creds.roleid" Property of the current secure session;

3596 AND the resource of the request matches one of the resources Property of the ACE;

3597 AND the ACE is currently valid.

3598 A request will match an ACE2 if any of the following are true:

3599 1) The ACE2 "subject" Property is of type oic.sec.didtype has a UUID value that matches the  
3600 "deviceuuid" Property associated with the secure session;

3601 AND the Resource of the request matches one of the resources Property of the ACE2  
3602 oic.sec.ace2.resource-ref;

3603 AND the ACE2 is currently valid.

3604 2) The ACE2 "subject" Property is of type oic.sec.conntype and has the wildcard value that  
3605 matches the currently established connection type;

3606 AND the resource of the request matches one of the resources Property of the ACE2  
3607 oic.sec.ace2.resource-ref;

3608 AND the ACE2 is currently valid.

3609 3) When Client authentication uses a certificate credential;

3610 AND one of the roleid values contained in the role certificate matches the roleid Property of  
3611 the ACE2 oic.sec.roletype;

3612 AND the role certificate public key matches the public key of the certificate used to establish  
3613 the current secure session;

3614 AND the resource of the request matches one of the array elements of the resources Property  
3615 of the ACE2 oic.sec.ace2.resource-ref;

3616 AND the ACE2 is currently valid.

3617 4) When Client authentication uses a certificate credential;

3618 AND the CoAP payload query string of the request specifies a role, which is member of the  
3619 set of roles contained in the role certificate;

3620 AND the roleid values contained in the role certificate matches the roleid Property of the  
3621 ACE2 oic.sec.roletype;

3622 AND the role certificate public key matches the public key of the certificate used to establish  
3623 the current secure session;

3624 AND the resource of the request matches one of the resources Property of the ACE2  
3625 oic.sec.ace2.resource-ref;

3626 AND the ACE2 is currently valid.

3627 5) When Client authentication uses a symmetric key credential;

3628 AND one of the roleid values associated with the symmetric key credential used in the secure  
3629 session, matches the roleid Property of the ACE2 oic.sec.roletype;

3630 AND the resource of the request matches one of the array elements of the resources Property  
3631 of the ACE2 oic.sec.ace2.resource-ref;

3632 AND the ACE2 is currently valid.

3633 6) When Client authentication uses a symmetric key credential;

3634 AND the CoAP payload query string of the request specifies a role, which is contained in the  
3635 "oic.r.cred.creds.roleid" Property of the current secure session;

3636 AND CoAP payload query string of the request specifies a role that matches the "roleid"  
3637 Property of the ACE2 oic.sec.roletype;

3638 AND the resource of the request matches one of the array elements of the "resources"  
3639 Property of the ACE2 oic.sec.ace2.resource-ref;

3640 AND the ACE2 is currently valid.

3641 A request is granted if ANY of the 'matching' ACEs contains the permission to allow the request.  
3642 Otherwise, the request is denied.

3643 There is no way for an ACE to explicitly deny permission to a resource. Therefore, if one Device  
3644 with a given role should have slightly different permissions than another Device with the same  
3645 role, they must be provisioned with different roles.

3646 The Server is required to verify that any hosted Resource has authorized access by the Client  
3647 requesting access. The "/oic/sec/acl2" Resource is co-located on the Resource host so that the  
3648 Resource request processing should be applied securely and efficiently. See Annex A for  
3649 example.

### 3650 **13.6 Access Manager ACL Resource**

3651 "oic.r.amacl" Resource is defined in Table 51.



3652

**Table 51 – 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

3653 Table 52 defines the Properties of "oic.r.amacl".

3654

**Table 52 – 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

3655 The AMS should be used to centralize management of access policy, but requires Servers to  
 3656 open a connection to the AMS whenever the named Resources are accessed. See A.2 for  
 3657 example.

3658 **13.7 Signed ACL Resource**

3659 "oic.r.sacl" Resource is defined in Table 53.

3660

**Table 53 – 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

3661 Table 54 defines the Properties of "oic.r.sacl".

3662

**Table 54 – 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 owneruuid 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 devowneruid 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

3663 Table 55 defines the Properties of "oic.sec.sigtype".

3664 **Table 55 – 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

3665 **13.8 Provisioning Status Resource**

3666 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning  
3667 should be Client-directed or Server-directed. Client-directed provisioning relies on a Client device  
3668 to determine what, how and when Server Resources should be instantiated and updated. Server-  
3669 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-  
3670 directed provisioning depends on configuration of the rowneruid Property of the "/oic/sec/doxm",  
3671 "/oic/sec/cred" and "/oic/sec/acl2" Resources to identify the device ID of the trusted DOTS, CMS  
3672 and AMS services respectively. Furthermore, the "/oic/sec/cred" Resource should be provisioned  
3673 at ownership transfer with credentials necessary to open a secure connection with appropriate  
3674 support service.

3675 "oic.r.pstat" Resource is defined in Table 56.

3676 **Table 56 – 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

3677 Table 57 defines the Properties of "oic.r.pstat".

Table 57 – 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	R		Current Mode
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes	RW		Target Mode
Operational Mode	om	oic.sec.pomtype	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.pomtype	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.

3679 The provisioning status Resource "/oic/sec/pstat" is used to enable Devices to perform self-  
3680 directed provisioning. Devices are aware of their current configuration status and a target  
3681 configuration objective. When there is a difference between current and target status, the Device

3682 should consult the rowneruid Property of "/oic/sec/cred" Resource to discover whether any  
 3683 suitable provisioning services exist. The Device should request provisioning if configured to do so.  
 3684 The om Property of "/oic/sec/pstat" Resource will specify expected Device behaviour under these  
 3685 circumstances.

3686 Self-directed provisioning enables Devices to function with greater autonomy to minimize  
 3687 dependence on a central provisioning authority that should be a single point of failure in the OCF  
 3688 Security Domain.

3689 Table 58 defines the Properties of "/oic/sec/dostype".

3690 **Table 58 – 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	UINT 16	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

3691 In all Device states:

3692 – An authenticated and authorised Client may change the Device state of a Device by updating  
 3693 pstat.dos.s to the desired value. The allowed Device state transitions are defined in  
 3694 Figure 27.

3695 – Prior to updating pstat.dos.s, the Client configures the Device to meet entry conditions for the  
 3696 new Device state. The SVR definitions define the entity (Client or Server) expected to perform  
 3697 the specific SVR configuration change to meet the entry conditions. Once the Client has  
 3698 configured the aspects for which the Client is responsible, it may update pstat.dos.s. The  
 3699 Server then makes any changes for which the Server is responsible, including updating  
 3700 required SVR values, and set pstat.dos.s to the new value.

3701 – The pstat.dos.p Property is read-only by all Clients.

3702 – The Server sets pstat.dos.p to TRUE before beginning the process of updating pstat.dos.s,  
 3703 and sets it back to FALSE when the pstat.dos.s change is completed.

3704 Any requests to update pstat.dos.s while pstat.dos.p is TRUE are denied.

3705 When Device state is RESET:

3706 – All SVR content is removed and reset to manufacturer default values.

3707 – The default manufacturer Device state is RESET.

3708 – NCRs are reset to manufacturer default values.

3709 – NCRs are inaccessible.

3710 – After successfully processing RESET the SRM transitions to RFOTM by setting s Property of  
3711 "/oic/sec/dostype" Resource to RFOTM.

3712 When Device state is RFOTM:

3713 – NCRs are inaccessible.

3714 – Before OTM is successful, the deviceuuid Property of "/oic/sec/doxm" Resource shall be set  
3715 to a temporary non-repeated value as defined in clauses 13.2 and 13.16.

3716 – Before OTM is successful, the s Property of "/oic/sec/dostype" Resource is read-only by  
3717 unauthenticated requestors

3718 – After the OTM is successful, the s Property of "/oic/sec/dostype" Resource is read-write by  
3719 authorized requestors.

3720 – The negotiated Device OC is used to create an authenticated session over which the DOTS  
3721 directs the Device state to transition to RFPRO.

3722 – If an authenticated session cannot be established the ownership transfer session should be  
3723 disconnected and SRM sets back the Device state to RESET state.

3724 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the  
3725 SRM asserts the OTM failed, should be disconnected, and transitions to RESET  
3726 (/pstat.dos.s=RESET).

3727 – The DOTS UPDATES the devowneruuid Property in the /doxm Resource to a non-nil UUID  
3728 value. The DOTS (or other authorized client) may update it multiple times while in RFOTM. It  
3729 is not updatable while in other device states except when the Device state returns to RFOTM  
3730 through RESET.

3731 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done,  
3732 the DOTS UPDATES the "owned" Property in the /doxm Resource to "true".

3733 When Device state is RFPRO:

3734 – The s Property of "/oic/sec/dostype" Resource is read-only by unauthorized requestors and  
3735 read-write by authorized requestors.

3736 – NCRs are inaccessible, except for Easy Setup Resources, if s supported.

3737 – The OCF Server may re-create NCRs.

3738 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.

3739 – An authorized Client may perform consistency checks on SVRs to determine which shall be  
3740 re-provisioned.

3741 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if  
3742 the Device has already transitioned from SRESET but consistency checks continue to fail.

3743 – The authorized Client sets the /pstat.dos.s=RFNOP.

3744 When Device state is RFNOP:

3745 – The /pstat.dos.s Property is read-only by unauthorized requestors and read-write by  
3746 authorized requestors.

3747 – NCRs, SVRs and core Resources are accessible following normal access processing.

3748 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or  
3749 RESET.

3750 When Device state is SRESET:

3751 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to  
3752 access or reference them.

- 3753 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
- 3754 include devowneruuid Property of the "/oic/sec/doxm" Resource,
- 3755 "creds":[{"subjectuid":<devowneruuid>},...] Property of the "/oic/sec/cred" Resource and
- 3756 s Property of the "/oic/sec/dostype" Resource of "/oic/sec/pstat" Resource.
- 3757 – The certificates that identify and authorize the Device owner are sufficient to re-create
- 3758 minimalist /cred and /doxm resources enabling Device owner control of SRESET. If the SRM
- 3759 can't establish these Resources, then it will transition to RESET state.
- 3760 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as
- 3761 needed to ensure they are available for continued provisioning in RFPRO or for normal
- 3762 functioning in RFNOP.
- 3763 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING
- 3764 dos.s Property of the /pstat Resource with RFPRO or RFNOP values
- 3765 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to
- 3766 Device owner privileges.
- 3767 – The SRM asserts a Client-directed operational mode (e.g. /pstat.om=CLIENT\_DIRECTED).
- 3768 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.
- 3769 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning
- 3770 mode without selecting any particular value.
- 3771 "oic.sec.dpmttype" is defined in Table 59.

3772 **Table 59 – 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

3773 Table 60 and Table 61 define the values of "oic.sec.dpmttype".

3774 **Table 60 – 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)

3775 **Table 61 – Value Definition of the "oic.sec.dpmttype" Property (High-Byte)**

Value	Device Mode	Description
bx0000,0000 – bx1111,1111	<Reserved>	Reserved for later use

3776 The *provisioning operation mode* type is an 8-bit mask enumerating the various provisioning  
3777 operation modes.

3778 "oic.sec.pomtype" is defined in Table 62.

3779 **Table 62 – 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

3780 Table 63 defines the values of "oic.sec.pomtype".

3781 **Table 63 – 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

3782 **13.9 Certificate Signing Request Resource**

3783 The "/oic/sec/csr" Resource is used by a Device to provide its desired identity, public key to be  
 3784 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC  
 3785 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of  
 3786 "/oic/sec/doxm" Resource has a 1 in the 0x8 bit position), the Device shall have a "/oic/sec/csr"  
 3787 Resource.

3788 "oic.r.csr" Resource is defined in Table 64.

3789 **Table 64 – 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

3790 Table 65 defines the Properties of "oic.r.csr".

3791 **Table 65 – 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
----------	----------	--------	---	-----	---

3792 The Device chooses which public key to use, and may optionally generate a new key pair for this  
3793 purpose.

3794 In the CSR, the Common Name component of the Subject Name shall contain a string of the  
3795 format "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122.  
3796 The Common Name, and other components of the Subject Name, may contain other data. If the  
3797 Device chooses to include additional information in the Common Name component, it shall delimit  
3798 it from the UUID field by white space, a comma, or a semicolon.

3799 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to  
3800 generate a new key pair, the Device may begin generation of a key pair as a result of a  
3801 RETRIEVE of this resource. If the Device cannot immediately respond to the RETRIEVE request  
3802 due to time required to generate a key pair, the Device shall return an "operation pending" error.  
3803 This indicates to the Client that the Device is not yet ready to respond, but will be able at a later  
3804 time. The Client should retry the request after a short delay.

### 3805 **13.10 Roles Resource**

3806 The roles Resource maintains roles that have been asserted with role certificates, as described in  
3807 clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role  
3808 certificate. Servers shall only grant access to the roles information associated with the public key  
3809 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.  
3810 See 10.4.2 for how role certificates are validated.

3811 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS  
3812 session with a Client, if is not already created. The roles Resource shall only expose a secured  
3813 OCF Endpoint in the "/oic/res" response. A Server shall retain the roles Resource at least as long  
3814 as the (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least  
3815 until the certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of  
3816 clause 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A  
3817 Server should regularly inspect the contents of the roles resource and purge contents based on a  
3818 policy it determines based on its resource constraints. For example, expired certificates, and  
3819 certificates from Clients that have not been heard from for some arbitrary period of time could be  
3820 candidates for purging.

3821 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In  
3822 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall  
3823 behave as follows. Unlisted operations are implementation specific and not reliable.

3824 1) A RETRIEVE request shall return all previously asserted roles associated with the currently  
3825 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query  
3826 parameter is not supported; all previously asserted roles associated with the currently  
3827 connected and authenticated Client's identity are returned.

3828 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties  
3829 included in the array as follows:

3830 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the  
3831 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid"  
3832 value.



3833 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,  
 3834 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed  
 3835 response and a duplicate entry shall not be added to the array.

3836 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored  
 3837 by the Server. The Server shall assign a unique "credid" value for every entry of the  
 3838 "roles" array.

3839 3) A DELETE request without a "credid" query parameter shall remove all entries from the  
 3840 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
 3841 Client's identity.

3842 4) A DELETE request with a "credid" query parameter shall remove only the entries of the  
 3843 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated  
 3844 Client's identity and where the corresponding "credid" matches the entry.

3845 NOTE The "oic.r.roles" Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined  
 3846 in ISO/IEC 30118-1:2018.

3847 "oic.r.roles" Resource is defined in Table 66.

3848 **Table 66 – 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

3849 Table 67 defines the Properties of "oic.r.roles".

3850 **Table 67 – 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

3851 Because "oic.r.roles" shares the "oic.sec.cred" schema with "oic.r.cred", "subjectuud" is a required Property. However,  
 3852 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property  
 3853 is contained in an UPDATE request to the "/oic/sec/roles" Resource.

3854 **13.11 Account Resource**

3855 The Account Resource specifies the Properties based on IETF RFC 6749 Access Token based  
 3856 account creation. The mechanism to obtain credentials is described in clause 7.5. The Account  
 3857 Resource is used for Device Registration. The Account Resource is instantiated on the OCF  
 3858 Cloud as "oic/sec/account" SVR and is used by cloud-enabled Devices to register with the OCF  
 3859 Cloud. It should be only accessible on a secure channel; non-secure channel should not be able  
 3860 access this Resource.

3861 During the Device Registration process, an OCF Cloud can provide a distinct URI of another OCF  
 3862 Cloud ("redirected-to" OCF Cloud). Both initial and redirected-to OCF Clouds are expected to  
 3863 belong to the same Vendor; they are assumed to have the same UUID and are assumed to have  
 3864 an out-of-band communication mechanism established. Device does not have to perform the  
 3865 Device Registration on the redirected-to OCF Cloud and the OCF Cloud may ignore such  
 3866 attempts. Redirected-to OCF Cloud is expected to accept the Access Token, provided to the  
 3867 Device by the initial OCF Cloud.

3868 The "di", "uid", "refreshtoken" and "accesstoken" Properties of the Account Resource should be  
 3869 securely stored as described in clause 15.

3870 The RETRIEVE operation on OCF Cloud's "/oic/sec/account" Resource is not allowed and the  
 3871 OCF Cloud is expected to reject all attempts to perform such operation.

3872 The UPDATE operation on the OCF Cloud's "/oic/sec/account" Resource behaves as follows:

- 3873 – A Device intending to register with the OCF Cloud shall send UPDATE with following  
 3874 Properties "di" ("di" Property Value of "/oic/d" Resource), and "accesstoken" as configured by  
 3875 the Mediator ("at" Property Value of "oic.r.coapcloudconf" Resource). The OCF Cloud verifies  
 3876 it is the same "accesstoken" which was assigned to the Mediator for the corresponding "di"  
 3877 Property Value. The "accesstoken" is the permission for the Device to access the OCF Cloud.  
 3878 If the "apn" was included when the Mediator UPDATED the "oic.r.coapcloudconf" Resource,  
 3879 the Device shall also include "authprovider" Property when registering with the OCF Cloud. If  
 3880 no "apn" is specified, then the "authprovider" Property shall not be included in the UPDATE  
 3881 request.
- 3882 – OCF Cloud returns "accesstoken", "uid", "refreshtoken", "expiresin" It may also return  
 3883 "redirecturi". Received "accesstoken" is to be treated by Device as an Access Token with  
 3884 "Bearer" token type as defined in IETF RFC 6750. This "accesstoken" shall be used for the  
 3885 following Account Session start using "oic/sec/session" SVR. Received "refreshtoken" is to be  
 3886 treated by Device as a Refresh Token as defined in IETF RFC 6749. The Device stores the  
 3887 OCF Cloud's Response values. If "redirecturi" is received, Device shall use received value as  
 3888 a new OCF Cloud URI instead of "cis" Property Value of "oic.r.coapcloudconf" Resource for  
 3889 further connections.

3890 The DELETE operation on the OCF Cloud's "/oic/sec/account" Resource should behave as  
 3891 follows:

- 3892 – To deregister with the OCF Cloud, a DELETE operation shall be sent with the "accesstoken"  
 3893 and either "uid", or "di" to be deregistered with the OCF Cloud. On DELETE with the OCF  
 3894 Cloud, the Device should also delete values internally stored. Once deregister with an OCF  
 3895 Cloud, Device can connect to any other OCF Cloud. Device deregistered need to go through  
 3896 the steps in 7.5 again to be registered with the OCF Cloud.

3897 " oic.r.account " Resource is defined in Table 68.

3898 **Table 68 – 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.base ine	Resource used for a device to add itself under a given credential	N/A

3899 Table 69 defines the Properties of "oic.r.account ".

3900 **Table 69 – 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	refresh token	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.

3901 **13.12 Account Session resource**

3902 The "/oic/sec/session" Resource hosted on the OCF Cloud is used for creating connections with  
 3903 the OCF Cloud subsequent to Device registration though "/oic/sec/account" Resource. The  
 3904 "/oic/sec/session" Resource requires the device ID, User ID and Access Token which are stored  
 3905 securely on the Device.

3906 The "/oic/sec/session" Resource is exposed by the OCF Cloud. It should be only accessible on a  
 3907 secure channel; non-secure channel cannot access this Resource.

3908 The RETRIEVE operation on OCF Cloud's "/oic/sec/session" Resource is not allowed and the  
 3909 OCF Cloud is expected to reject all attempts to perform such operation.

3910 The UPDATE operation is defined as follows for OCF Cloud's "/oic/sec/session" Resource:

- 3911 – The Device connecting to the OCF Cloud shall send an UPDATE request message to the OCF  
 3912 Cloud's "/oic/sec/session" Resource. The message shall include the "di" Property Value of  
 3913 "/oic/d" Resource and "uid", "login" Value ("true" to establish connection; "false" to disconnect)  
 3914 and "accesstoken" as returned by OCF Cloud during Device Registration. The OCF Cloud  
 3915 verifies it is the same Access Token which was returned to the Device during Device  
 3916 Registration process. If Device was attempting to establish the connection and provided  
 3917 values were verified as correct by the OCF Cloud, OCF Cloud sends a response with  
 3918 remaining lifetime of the associated Access Token ("expiresin" Property Value).

3919 "/oic.r.session" Resource is defined in Table 70.

3920 **Table 70 – 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.base line	Resource that enables a device to manage its session using login or logout	N/A

3921 Table 71 defines the Properties of "oic.r.session".

3922 **Table 71 – 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

3923 **13.13 Account Token Refresh Resource**

3924 The "/oic/sec/tokenrefresh" Resource is used by the Device for refreshing the Access Token.

3925 The "/oic/sec/tokenrefresh" Resource is hosted by the OCF Cloud. It should be only accessible  
3926 on a secure channel; non-secure channel cannot access this Resource.

3927 The Device should use "/oic/sec/tokenrefresh" to refresh the Access Token with the OCF Cloud,  
3928 when the time specified in "expiresin" is near.

3929 The RETRIEVE operation on OCF Cloud's "/oic/sec/ tokenrefresh" Resource is not allowed and  
3930 the OCF Cloud is expected to reject all attempts to perform such operation.

3931 The UPDATE operation is defined as follows for "/oic/sec/tokenrefresh" Resource

3932 – The Device attempting to refresh the Access Token shall send an UPDATE request message  
3933 to the OCF Cloud's "/oic/sec/tokenrefresh" Resource. The message shall include the "di"  
3934 Property Value of "/oic/d" Resource, "uid" and "refreshtoken", as returned by OCF Cloud.

3935 – OCF Cloud response is expected to include a "refreshtoken", new "accesstoken", and  
3936 "expiresin". Received "accesstoken" is to be treated by Device as an Access Token with  
3937 "Bearer" token type as defined in IETF RFC 6750. This Access Token is the permission for  
3938 the Device to access the OCF Cloud. Received "refreshtoken" is to be treated by Device as a  
3939 Refresh Token as defined in IETF RFC 6749. Received "refreshtoken" may be the new  
3940 Refresh Token or the same one as provided by the Device in the UPDATE request. In case  
3941 when new distinct "refreshtoken" is provided by the OCF Cloud, the Device shall discard the  
3942 old value. The OCF Cloud's response values "refreshtoken", "accesstoken" and "expiresin" are  
3943 securely stored on the Device.

3944 "/oic.r.tokenrefresh" Resource is defined in Table 72.

3945 **Table 72 – 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

3946 Table 73 defines the Properties of "oic.r.tokenrefresh".

**Table 73 – 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)

### 3948 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3949 The SVRs expose the security-related Properties of the Device.

3950 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to  
3951 unauthenticated (anonymous) Clients could create privacy or security concerns.

3952 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for  
3953 the "oic.r.doxm" Resource to anonymous requesters, so that the Device can be discovered and  
3954 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the "oic.r.doxm"  
3955 Resource to anonymous requesters, to preserve privacy.

### 3956 **13.15 SVRs, Discoverability and OCF Endpoints**

3957 All implemented SVRs shall be "discoverable" (reference ISO/IEC 30118-1:2018, Policy  
3958 Parameter clause 7.8.2.1.2).

3959 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS)  
3960 (reference ISO/IEC 30118-1:2018, clause 10).

3961 The "/oic/sec/doxm" Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM  
3962 (reference ISO/IEC 30118-1:2018, clause 10).

### 3963 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

#### 3964 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

3965 Unique identifiers are a privacy consideration due to their potential for being used as a tracking  
3966 mechanism. These include the following Resources and Properties:

- 3967 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 3968 – "/oic/p" Resource containing the "pi" Property.
- 3969 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

3970 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous  
3971 requestors. This implies any Client Device, including those with malicious intent, are able to  
3972 reliably obtain identifiers useful for building a log of activity correlated with a specific Platform  
3973 and Device.

3974 There are two strategies for privacy protection of Devices:

- 3975 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers  
3976 2) Limit identifier persistence to make it impractical for tracking use.  
3977 Both techniques can be used effectively together to limit exposure to privacy attacks.

- 3978 1) A Platform / Device manufacturer should specify a default ACL policy that restricts  
3979 anonymous requestors from accessing unique identifiers. An OCF Security Domain owner  
3980 should modify the ACL policy to grant access to authenticated Devices who, presumably, do  
3981 not present a privacy threat.
- 3982 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the  
3983 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and  
3984 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated  
3985 identifiers shall be:
- 3986 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
  - 3987 b) Generated by a function that is pre-image resistant, second pre-image resistant and  
3988 collision resistant

3989 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier  
3990 used to begin the onboarding process. However, attackers could obtain the value too and use it  
3991 for Device tracking throughout the Device's lifetime.

3992 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the  
3993 deviceuuid Property of the "/oic/sec/doxm" Resource to unauthenticated "/oic/res" and  
3994 "/oic/sec/doxm" Resource RETRIEVE requests when the devowneruuid Property of  
3995 "/oic/sec/doxm" Resource is the nil-UUID. The Server shall expose a new temporary non-  
3996 repeated deviceuuid Property of the "/oic/sec/doxm" Resource when the device state transitions  
3997 to RESET. This ensures the deviceuuid Property of the "/oic/sec/doxm" cannot be used to track  
3998 across multiple owners.

3999 The devowneruuid Property of "/oic/sec/doxm" Resource is initialized to the nil-UUID upon  
4000 entering RESET; which is retained until being set to a non-nil-UUID value during RFOTM device  
4001 state. The device shall supply a temporary, non-repeated deviceuuid Property of "/oic/sec/doxm"  
4002 Resource to RETRIEVE requests on "/oic/sec/doxm" and "/oic/res" Resources while  
4003 devowneruuid Property of "/oic/sec/doxm" Resource is the nil-UUID. During the OTM process the  
4004 DOTS shall UPDATE devowneruuid Property of the "/oic/sec/doxm" Resource to a non-nil UUID  
4005 value which is the trigger for the Device to expose its persistent or semi-persistent device  
4006 identifier. Therefore, the Device shall supply deviceuuid Property of "/oic/sec/doxm" Resource in  
4007 response to RETRIEVE requests while the devowneruuid Property of the "/oic/sec/doxm"  
4008 Resource is a non-nil-UUID value.

4009 The DOTS or AMS may also provision an ACL policy that restricts access to the "/oic/sec/doxm"  
4010 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent  
4011 device identifier via the deviceuuid Property value of the "/oic/sec/doxm" Resource.

4012 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent  
4013 or semi-persistent identifier using the "/oic/sec/cred" Resource to first establish an authenticated  
4014 connection. This is achieved by first provisioning a "/oic/sec/cred" Resource entry that contains  
4015 the Server's deviceuuid Property value of the "/oic/sec/doxm" Resource.

4016 The di Property in the "/oic/d" Resource shall mirror that of the deviceuuid Property of the  
4017 "/oic/sec/doxm" Resource. The DOTS should provision an ACL policy that restricts access to the  
4018 "/oic/d" resource such that only authenticated Clients are able to obtain the di Property of "/oic/d"  
4019 Resource. See clause 13.1 for deviceuuid Property lifecycle requirements.

4020 Servers should expose a temporary, non-repeated, piid Property of "/oic/p" Resource Value upon  
4021 entering RESET Device state. Servers shall expose a persistent value via the piid Property of

4022 "/oic/p" Property when the DOTS sets devowneruid Property to a non-nil-UUID value. An ACL  
 4023 policy on the "/oic/d" Resource should protect the piid Property of "/oic/p" Resource from being  
 4024 disclosed to unauthenticated requestors.

4025 Servers shall expose a temporary, non-repeated, pi Property value upon entering RESET Device  
 4026 state. Servers shall expose a persistent or semi-persistent platform identifier value via the pi  
 4027 Property of the "/oic/p" Resource when onboarding sets devowneruid Property to a non-nil-UUID  
 4028 value. An ACL policy on the "/oic/p" Resource should protect the pi Property from being disclosed  
 4029 to unauthenticated requestors.

4030 Table 74 depicts Core Resource Properties Access Modes given various Device States.

4031 **Table 74 – 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.

4032 Four identifiers are thought to be privacy sensitive:

- 4033 – "/oic/d" Resource containing the "di" and "piid" Properties.
- 4034 – "/oic/p" Resource containing the "pi" Property.
- 4035 – "/oic/sec/doxm" Resource containing the "deviceuuid" Property.

4036 There are three strategies for privacy protection of Devices:

- 4037 1) Apply access control to restrict read access to Resources containing unique identifiers. This  
 4038 ensures privacy sensitive identifiers do not leave the Device.
- 4039 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy  
 4040 sensitive identifiers are less effective for tracking and correlation.
- 4041 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can  
 4042 do so.

4043 These techniques can be used to limit exposure to privacy attacks. For example:

- 4044 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent  
 4045 identifiers can be created.
- 4046 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to  
 4047 facilitate onboarding.
- 4048 – Persistent and semi-persistent identifiers can be encrypted before sending them to another  
 4049 Device.

4050 A temporary, non-repeated identifier shall be:

- 4051 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
- 4052 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision  
4053 resistant

4054 NOTE This requirement is met through a vendor attestation certification mechanism.

### 4055 **13.16.2 Privacy Protecting the Device Identifiers**

4056 The "di" Property Value of the "/oic/d" Resource shall mirror that of the "deviceuuid" Property of  
4057 the "/oic/sec/doxm" Resource. The Device should use a new, temporary non-repeated identifier in  
4058 place of the "deviceuuid" Property Value of "/oic/sec/doxm" Resource upon entering the RESET  
4059 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID  
4060 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value  
4061 of the "/oic/sec/doxm" Resource after the DOTS sets the "devowneruuid" Property to a non-nil-  
4062 UUID value. The temporary identifier should not change more frequently than once per Device  
4063 state transition to RESET.

4064 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4065 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"  
4066 Property values:
  - 4067 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di")  
4068 Property value only if responding to an authenticated requestor and the "deviceuuid" (or  
4069 "di") value is confidentiality protected.
  - 4070 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if  
4071 responding to an unauthenticated requestor.
- 4072 – The AMS should provision an ACL policy on the "/oic/sec/doxm" and "/oic/d" resources to  
4073 further protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

4074 See 13.2 for deviceuuid Property lifecycle requirements.

4075 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary  
4076 discovery requests. This is achieved by provisioning a "/oic/sec/cred" Resource entry that contains the Server's  
4077 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

### 4078 **13.16.3 Privacy Protecting the Protocol Independent Device Identifier**

4079 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property  
4080 Value of "/oic/d" Resource upon entering the RESET Device state. If a temporary, non-repeated  
4081 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID  
4082 value. The Device should use its persistent "piid" Property value after the DOTS sets the  
4083 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change  
4084 more frequently than once per Device state transition to RESET.

4085 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4086 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
  - 4087 – The Device should include its persistent "piid" Property value only if responding to an  
4088 authenticated requestor and the "piid" value is confidentiality protected.
  - 4089 – The Device should include a temporary non-repeated "piid" Property value if responding to  
4090 an unauthenticated requestor.
- 4091 – The AMS should provision an ACL policy on the "/oic/d" Resource to further protect the piid  
4092 Property of "/oic/p" Resource from being disclosed unnecessarily.



4093 **13.16.4 Privacy Protecting the Platform Identifier**

4094 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property  
4095 Value of the "/oic/p" Resource upon entering the RESET Device state. This value should be  
4096 exposed while the "devowneruid" Property has a nil UUID value. The Device should use its  
4097 persistent (or semi-persistent) "pi" Property value after the DOTS sets the "devowneruid"  
4098 Property to a non-nil-UUID value. The temporary identifier should not change more frequently  
4099 than once per Device state transition to RESET.

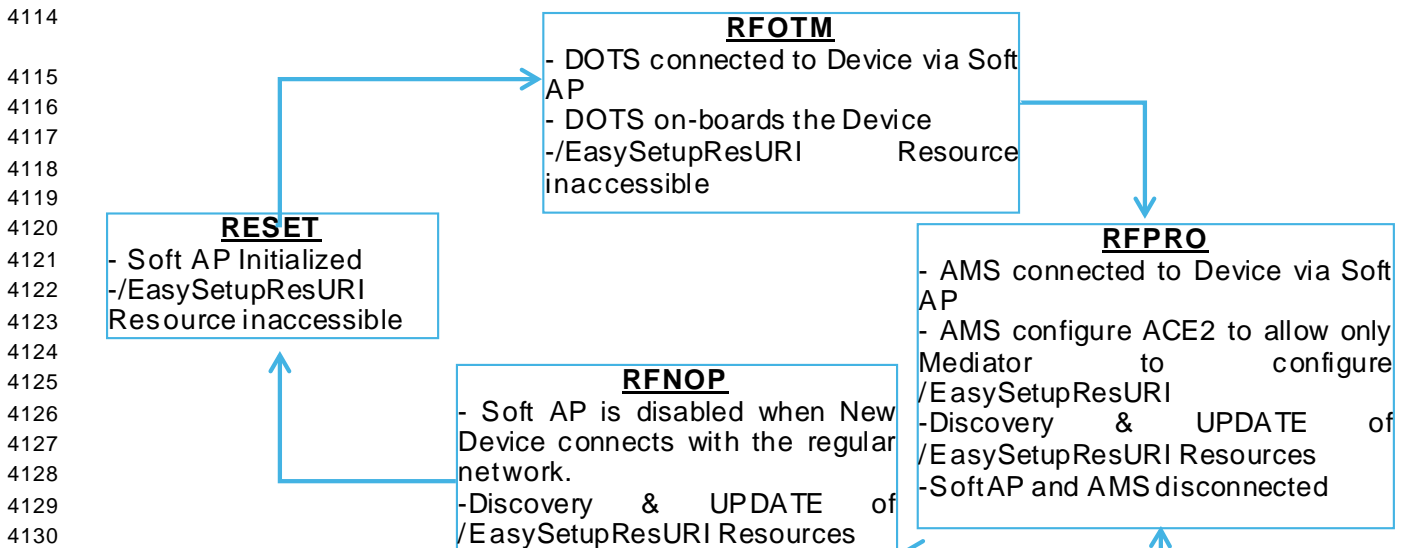
4100 Subsequent to the "devowneruid" being UPDATED to a non-nil UUID:

- 4101 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
  - 4102 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
  - 4103 responding to an authenticated requestor and the "pi" value is confidentiality protected.
  - 4104 – The Device should include a temporary non-repeated "pi" Property value if responding to
  - 4105 an unauthenticated requestor.
- 4106 – The AMS should provision an ACL policy on the "/oic/p" Resource to protect the pi Property
- 4107 from being disclosed unnecessarily.

4108 **13.17 Easy Setup Resource Device State**

4109 This clause only applies to a new Device that uses Easy Setup for ownership transfer as defined  
4110 in OCF Wi-Fi Easy Setup. Easy Setup has no impact to new Devices that have a different way of  
4111 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup  
4112 Devices.

4113 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



4131 **Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states**

4132 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's  
4133 state.

4134 While it is reasonable for a user to expect that power cycling a new Device will turn on the Soft  
4135 AP for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it  
4136 is a security risk to make this the default behaviour of a device that remains unenrolled beyond a  
4137 reasonable period after first boot.

4138 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

- 4139 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes  
4140 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for  
4141 Easy Setup.
- 4142 – If a new Device tried and failed to complete Easy Setup Enrolment immediately following the  
4143 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically  
4144 for another 30 minutes upon being power cycled, provided that the power cycle occurs within  
4145 3 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup  
4146 Soft AP directly without a factory reset, it is not necessary to turn it back on if it was on  
4147 immediately prior to power cycle, because the user obviously knows how to initiate the  
4148 process manually.
- 4149 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft  
4150 AP should not turn back on for Easy Setup until another factory reset occurs, or the user  
4151 initiates the Easy Setup Soft AP directly.
- 4152 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the new  
4153 Device to connect to the Enroller.
- 4154 – The Easy Setup Soft AP shall be disabled when the new Device successfully connects to the  
4155 Enroller.
- 4156 – Once a new Device has successfully connected to the Enroller, it shall not turn the Easy  
4157 Setup Soft AP back on for Easy Setup Enrolment again unless the Device is factory reset, or  
4158 the user initiates the Easy Setup Soft AP directly.
- 4159 – Just Works OTM shall not be enabled on Devices which support Easy Setup.
- 4160 – The Soft AP shall be secured (e.g. shall not expose an open AP).
- 4161 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase  
4162 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed on  
4163 a label, sticker, packaging etc., and may be entered by the user into the Mediator device.
- 4164 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the  
4165 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by  
4166 an attacker with knowledge of the Device type, model, manufacturer, or any other information  
4167 discoverable through Device's exposed interfaces.
- 4168 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the  
4169 "/example/WiFiConfResURI" Resource), for potential selection by the Mediator in connecting the  
4170 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller,  
4171 for use in connecting the Enrollee to the Enroller.
- 4172 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the  
4173 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.
- 4174 The "/example/EasySetupResURI" Resource should not be discoverable in RFOTM or SRESET  
4175 state. After ownership transfer process is completed with the DOTS, and the Device enters in  
4176 RFPRO Device state, the "/example/EasySetupResURI" may be Discoverable. The DOTS may be  
4177 hosted on the Mediator Device.
- 4178 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership  
4179 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used  
4180 by AMS for "/oic/sec/acl2" Resource provisioning in RFPRO state. The CoAPS session  
4181 authentication and encryption is already defined in the Security spec.
- 4182 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following  
4183 Resources to be only configurable by the Mediator Device with permission to UPDATE or  
4184 RETRIEVE access:
- 4185 – /example/EasySetupResURI

4186 – /example/WifiConfResURI  
4187 – /example/DevConfResURI  
4188 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

```
4189 {  
4190     "subject": { "uuid": "<insert-UUID-of-Mediator>" },  
4191     "resources": [  
4192         { "href": "/example/EasySetupResURI" },  
4193         { "href": "/example/WiFiConfResURI" },  
4194         { "href": "/example/DevConfResURI" },  
4195     ],  
4196     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)  
4197 }
```

4198 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to  
4199 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

4200 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE  
4201 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

## 4202 **14 Security Hardening Guidelines/ Execution Environment Security**

### 4203 **14.1 Preamble**

4204 This is an informative clause. Many TGs in OCF have security considerations for their protocols  
4205 and environments. These security considerations are addressed through security mechanisms  
4206 specified in the security documents for OCF. However, effectiveness of these mechanisms  
4207 depends on security robustness of the underlying hardware and software Platform. This clause  
4208 defines the components required for execution environment security.

### 4209 **14.2 Execution Environment Elements**

#### 4210 **14.2.1 Execution Environment Elements General**

4211 Execution environment within a computing Device has many components. To perform security  
4212 functions in a robustness manner, each of these components has to be secured as a separate  
4213 dimension. For instance, an execution environment performing AES cannot be considered secure  
4214 if the input path entering keys into the execution engine is not secured, even though the  
4215 partitions of the CPU, performing the AES encryption, operate in isolation from other processes.  
4216 Different dimensions referred to as elements of the execution environment are listed below. To  
4217 qualify as a secure execution environment (SEE), the corresponding SEE element must qualify as  
4218 secure.

- 4219 – (Secure) Storage
- 4220 – (Secure) Execution engine
- 4221 – (Trusted) Input/output paths
- 4222 – (Secure) Time Source/clock
- 4223 – (Random) number generator
- 4224 – (Approved) cryptographic algorithms
- 4225 – Hardware Tamper (protection)

4226 NOTE Software security practices (such as those covered by OWASP) are outside scope of this document, as  
4227 development of secure code is a practice to be followed by the open source development community. This document  
4228 will however address the underlying Platform assistance required for executing software. Examples are secure boot  
4229 and secure software upgrade.

4230 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,  
4231 14.2.7.

#### 4232 **14.2.2 Secure Storage**

##### 4233 **14.2.2.1 Secure Storage General**

4234 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive  
4235 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,  
4236 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive  
4237 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both  
4238 its integrity and confidentiality be maintained.

4239 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive  
4240 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either  
4241 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication  
4242 and encryption, it must maintain its integrity against intentional or accidental alteration.

4243 A partial list of Sensitive Data is outlined in Table 75:

**Table 75 – 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

4245 Exact method of protection for secure storage is implementation specific, but typically  
4246 combinations of hardware and software methods are used.

#### 4247 **14.2.2.2 Hardware Secure Storage**

4248 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric  
4249 and asymmetric private keys, access credentials, and personal private data. Hardware secure  
4250 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes  
4251 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

4252 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides  
4253 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or  
4254 electronic attacks. It is not necessary to prevent the attacks themselves, but an attempted attack  
4255 should not result in an unauthorized entity successfully retrieving Sensitive Data.

4256 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data  
4257 from attacks that include but are not limited to:

- 4258 1) Physical decapping of chip packages to optically read NVRAM contents
- 4259 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 4260 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit  
4261 patterns of Critical Sensitive Data
- 4262 4) Use of malicious software or firmware to read memory contents at rest or in transit within a  
4263 microcontroller
- 4264 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

#### 4265 **14.2.2.3 Software Storage**

4266 It is generally NOT recommended to rely solely on software and unsecured memory to store  
4267 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and  
4268 encryption keys should be housed in hardware secure storage whenever possible.

4269 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable  
4270 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

#### 4271 **14.2.2.4 Additional Security Guidelines and Best Practices**

4272 Some general practices that can help ensure that Sensitive Data is not compromised by various  
4273 forms of security attacks:

- 4274 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG  
4275 used for authentication challenges can substantially degrade security strength. For this  
4276 reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source  
4277 be used for all authentication challenges.
- 4278 2) Secure download and boot – To prevent the loading and execution of malicious software,  
4279 where it is practical, it is recommended that Secure Download and Secure Boot methods that  
4280 authenticate a binary's source as well as its contents be used.
- 4281 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered  
4282 insecure and shall not be used for any security-related function:
  - 4283 a) SHA-1
  - 4284 b) MD5
  - 4285 c) RC4
  - 4286 d) RSA 1024
- 4287 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is  
4288 stored in Secure Storage, any use of that data that requires its transmission out of that  
4289 Secure Storage should be encrypted to prevent eavesdropping by malicious software within  
4290 an MCU/MPU.
- 4291 5) It is recommended to avoid using wildcard in Subject Id ("\*"), when setting up "oic.r.cred"  
4292 Resource entries, since this opens up an identity spoofing opportunity.
- 4293 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device  
4294 meets security requirements for its intended uses. As an example, IoTivity is a reference  
4295 implementation intended to be used as a basis for a product, but IoTivity has not undergone  
4296 3rd party security review, penetration testing, etc. Any Device based on IoTivity should  
4297 undergo appropriate penetration testing and security review prior to sale or deployment.
- 4298 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to  
4299 consumers. Changes should be made to a public and accessible website. Expectations  
4300 should be clear as to what will be supported and for how long the Device vendor expects to  
4301 support security updates to the software, operating system, drivers, networking, firmware and  
4302 hardware of the device.
- 4303 8) Device vendor has not implemented test or debug interfaces on the Device which are  
4304 operable or which can be enabled which might present an attack vector on the Device which  
4305 circumvents the interface-level security or access policies of the Device.
- 4306 9) Device vendor understands that if an application running on the Device has access to  
4307 cryptographic elements such as the private keys or Ownership Credential, then those  
4308 elements have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or  
4309 a Device with access to the Internet beyond the local network, the execution of critical  
4310 functions should take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4311 10) Any PINs or fixed passphrases used for onboarding, Wi-Fi Easy Setup, SoftAP management  
4312 or access, or other security-critical function, should be sufficiently unique (do not duplicate  
4313 passphrases. The creation of these passphrases or PINS should not be algorithmically  
4314 deterministic nor should they use insufficient entropy in their creation.
- 4315 11) Ensure that there are no remaining "VENDOR\_TODO" items in the source code.

4316 12) If the implementation of this document uses the "Just Works" onboarding method, understand  
4317 that there is a man-in-the-middle vulnerability during the onboarding process where a  
4318 malicious party could intercept messages between the device being onboarded and the OBT  
4319 and could persist, acting as an intermediary with access to message traffic, during the lifetime  
4320 of that onboarded device. The recommended best practice would be to use an alternate  
4321 ownership transfer method (OTM) instead of "Just Works".

4322 13) It is recommended that at least one static and dynamic analysis tool<sup>1</sup> be applied to any  
4323 proposed major production release of the software before its release, and any vulnerabilities  
4324 resolved.

4325 14) To avoid a malicious device being able to covertly join an OCF Security Domain,  
4326 implementers of any OBT may eliminate completely autonomous sequences where a device  
4327 is brought into the OCF Security Domain without any authorization by the owner. Consider  
4328 either including a confirmation with the OCF Security Domain owner/operator (e.g. "Do you  
4329 want to add 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'? Yes/No/Cancel?") or a  
4330 confirmation with a security policy (e.g. an enterprise policy where the OCF Security Domain  
4331 admin can bulk-onboard devices).

### 4332 **14.2.3 Secure execution engine**

4333 Execution engine is the part of computing Platform that processes security functions, such as  
4334 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine  
4335 requires the following

4336 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This  
4337 includes isolation of CPU caches, and all of execution elements that needed to be considered  
4338 as part of trusted (crypto) boundary.

4339 – Isolation of data paths into and out of execution engine. For instance, both unencrypted but  
4340 sensitive data prior to encryption or after decryption, or cryptographic keys used for  
4341 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

### 4342 **14.2.4 Trusted input/output paths**

4343 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be  
4344 protected. This includes paths into and out secure execution engine and secure memory.

4345 Path protection can be both hardware based (e.g. use of a privileged bus) or software based  
4346 (using encryption over an untrusted bus).

### 4347 **14.2.5 Secure clock**

4348 Many security functions depend on time-sensitive credentials. Examples are time stamped  
4349 Kerberos tickets, OAuth tokens, X.509 certificates, OSCP response, software upgrades, etc.  
4350 Lack of secure source of clock can mean an attacker can modify the system clock and fool the  
4351 validation mechanism. Thus an SEE needs to provide a secure source of time that is protected  
4352 from tampering. Trustworthiness from security robustness standpoint is not the same as accuracy.  
4353 Protocols such as NTP can provide rather accurate time sources from the network, but are not  
4354 immune to attacks. A secure time source on the other hand can be off by seconds or minutes  
4355 depending on the time-sensitivity of the corresponding security mechanism. Secure time source  
4356 can be external as long as it is signed by a trusted source and the signature validation in the  
4357 local Device is a trusted process (e.g. backed by secure boot).

### 4358 **14.2.6 Approved algorithms**

4359 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and  
4360 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by  
4361 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only

---

<sup>1</sup> A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4362 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic  
4363 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are  
4364 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms  
4365 (even if they deemed stronger by some parties) must be considered non-approved.

4366 The set of algorithms to be considered for approval are algorithms for

- 4367 – Hash functions
- 4368 – Signature algorithms
- 4369 – Encryption algorithms
- 4370 – Key exchange algorithms
- 4371 – Pseudo Random functions (PRF) used for key derivation

4372 This list will be included in this or a separate security robustness rules document and must be  
4373 followed for all security specifications within OCF.

#### 4374 **14.2.7 Hardware tamper protection**

4375 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not  
4376 requirements) regarding tamper protection for cryptographic module

- 4377 – Production-grade (lowest level): this means components that include conformal sealing  
4378 coating applied over the module's circuitry to protect against environmental or other physical  
4379 damage. This does not however require zeroization of secret material during physical  
4380 maintenance. This definition is borrowed from FIPS 140-2 security level 1.
- 4381 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,  
4382 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from  
4383 FIPS 140-2 security level 2.
- 4384 – Tamper resistance (highest level), this means there is a response to physical tempering that  
4385 typically includes zeroization of sensitive material on the module. This definition is borrowed  
4386 from FIPS 140-2 security level 3.

4387 It is difficult of specify quantitative certification test cases for accreditation of these levels.  
4388 Content protection regimes usually talk about different tools (widely available, specialized and  
4389 professional tools) used to circumvent the hardware protections put in place by manufacturing. If  
4390 needed, OCF can follow that model, if and when OCF engage in distributing sensitive key  
4391 material (e.g. PKI) to its members.

### 4392 **14.3 Secure Boot**

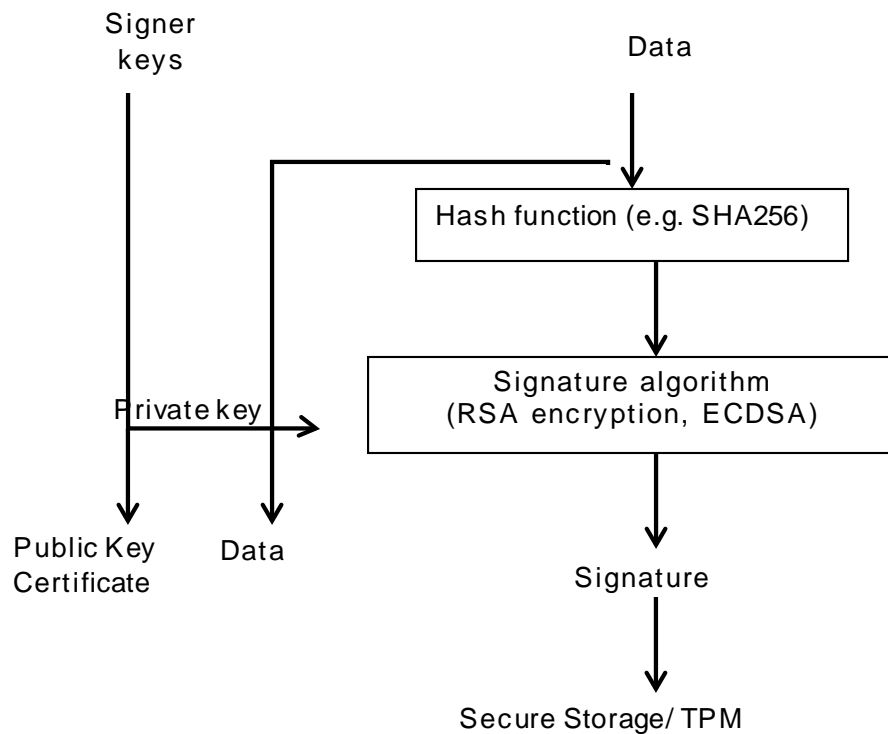
#### 4393 **14.3.1 Concept of software module authentication**

4394 In order to ensure that all components of a Device are operating properly and have not been  
4395 tampered with, it is best to ensure that the Device is booted properly. There may be multiple  
4396 stages of boot. The end result is an application running on top an operating system that takes  
4397 advantage of memory, CPU and peripherals through drivers.

4398 The general concept is that each software module is invoked only after cryptographic integrity  
4399 verification is complete. The integrity verification relies on the software module having been  
4400 hashed (e.g. SHA\_1, SHA\_256) and then signed with a cryptographic signature algorithm with  
4401 (e.g. RSA), with a key that only a signing authority has access to.

4402 Figure 40 depicts software module authentication.

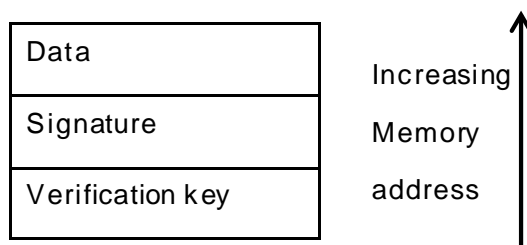




**Figure 40 – Software Module Authentication**

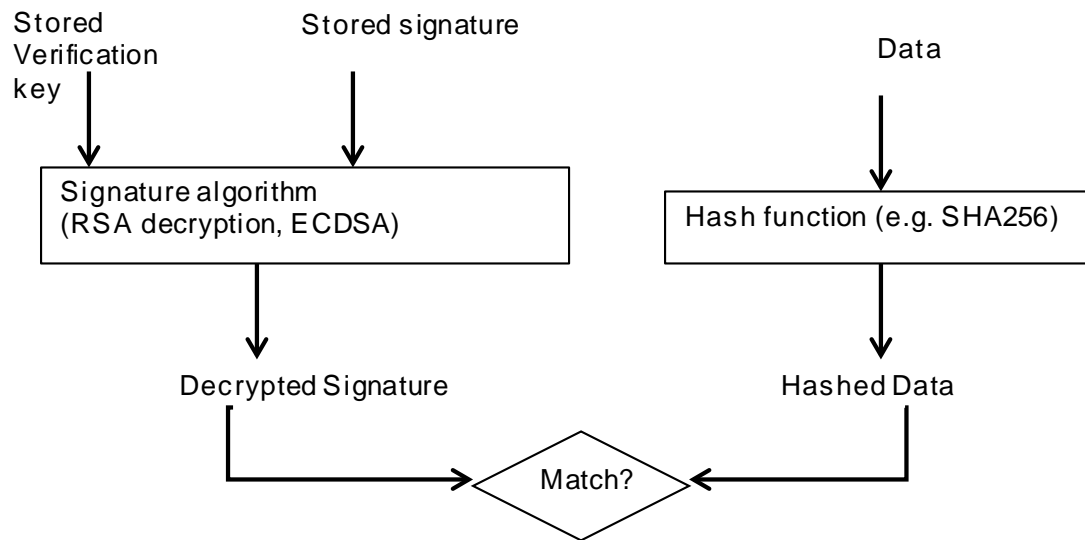
4403  
 4404 After the data is signed with the signer’s signing key (a private key), the verification key (the  
 4405 public key corresponding to the private signing key) is provided for later verification. For lower  
 4406 level software modules, such as bootloaders, the signatures and verification keys are inserted  
 4407 inside tamper proof memory, such as one-time programmable memory or TPM. For higher level  
 4408 software modules, such as application software, the signing is typically performed according to  
 4409 the PKCS#7 format IETF RFC 2315, where the signedData format includes both indications for  
 4410 signature algorithm, hash algorithm as well as the signature verification key (or certificate).  
 4411 Secure boot does not require use of PKCS#7 format.

4412 Figure 41 depicts verification software module.



**Figure 41 – Verification Software Module**

4413  
 4414 As shown in Figure 42. the verification module first decrypts the signature with the verification  
 4415 key (public key of the signer). The verification module also calculates a hash of the data and then  
 4416 compares the decrypted signature (the original) with the hash of data (actual) and if the two  
 4417 values match, the software module is authentic.



4418 **Figure 42 – Software Module Authenticity**

4418

4419 **14.3.2 Secure Boot process**

4420 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/  
 4421 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to  
 4422 find out where the boot code is and then run the boot code (second-stage boot loader). The  
 4423 second stage bootloader is typically the process that loads the operating system (Kernel) and  
 4424 transfers the execution to the where the Kernel code is. Once the Kernel starts, it may load  
 4425 external Kernel modules and drivers.

4426 When performing a secure boot, it is required that the integrity of each boot loader is verified  
 4427 before executing the boot loader stage. As mentioned, while the signature and verification key for  
 4428 the lowest level bootloader is typically stored in tamper-proof memory, the signature and  
 4429 verification key for higher levels should be embedded (but attached in an easily accessible  
 4430 manner) in the data structures software.

4431 **14.3.3 Robustness Requirements**

4432 **14.3.3.1 Robustness General**

4433 To qualify as high robustness secure boot process, the signature and hash algorithms shall be  
 4434 one of the approved algorithms, the signature values and the keys used for verification shall be  
 4435 stored in secure storage and the algorithms shall run inside a secure execution environment and  
 4436 the keys shall be provided the SEE over trusted path.

4437 **14.3.3.2 Next steps**

4438 Develop a list of approved algorithms and data formats

4439 **14.4 Attestation**

4440 **14.5 Software Update**

4441 **14.5.1 Overview:**

4442 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;  
 4443 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and  
 4444 end-of-life stages for the Device remain outstanding. It is possible for the Device to require

4445 update during any of these stages, although the most likely times are during onboarding, regular  
4446 operation and maintenance. The aspects of the software include, but are not limited to, firmware,  
4447 operating system, networking stack, application code, drivers, etc.

#### 4448 **14.5.2 Recognition of Current Differences**

4449 Different manufacturers approach software update utilizing a collection of tools and strategies:  
4450 over-the-air or wired USB connections, full or partial replacement of existing software, signed and  
4451 verified code, attestation of the delivery package, verification of the source of the code, package  
4452 structures for the software, etc.

4453 It is recommended that manufacturers review their processes and technologies for compliance  
4454 with industry best-practices that a thorough security review of these takes place and that periodic  
4455 review continue after the initial architecture has been established.

4456 This document applies to software updates as recommended to be implemented by Devices; it  
4457 does not have any bearing on the above-mentioned alternative proprietary software update  
4458 mechanisms.

#### 4459 **14.5.3 Software Version Validation**

4460 Setting the Initiate Software Version Validation bit in the "/oic/sec/pstat.tm" Property (see  
4461 Table 57 defines the Properties of "oic.r.pstat".

4462 Table 57 of 13.8) indicates a request to initiate the software version validation process, the  
4463 process whereby the Device validates the software (including firmware, operating system, Device  
4464 drivers, networking stack, etc.) against a trusted source to see if, at the conclusion of the check,  
4465 the software update process will need to be triggered (see clause 14.5.4). When the Initiate  
4466 Software Version Validation bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged  
4467 Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and  
4468 initiates a software version check. Once the Device has determined if an update is available, it  
4469 sets the Initiate Software Version Validation bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE) if  
4470 an update is available or 0 (FALSE) if no update is available. To signal completion of the  
4471 Software Version Validation process, the Device sets the Initiate Software Version Validation bit  
4472 in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Software Version Validation  
4473 bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the validation process.

#### 4474 **14.5.4 Software Update**

4475 Setting the Initiate Secure Software Update bit in the "/oic/sec/pstat.tm" Property (see Table 57  
4476 defines the Properties of "oic.r.pstat".

4477 Table 57 of 13.8) indicates a request to initiate the software update process. When the Initiate  
4478 Secure Software Update bit of "/oic/sec/pstat.tm" is set to 1 (TRUE) by a sufficiently privileged  
4479 Client, the Device sets the "/oic/sec/pstat.cm" Initiate Software Version Validation bit to 0 and  
4480 initiates a software update process. Once the Device has completed the software update process,  
4481 it sets the Initiate Secure Software Update bit in the "/oic/sec/pstat.cm" Property to 1 (TRUE)  
4482 if/when the software was successfully updated or 0 (FALSE) if no update was performed. To  
4483 signal completion of the Secure Software Update process, the Device sets the Initiate Secure  
4484 Software Update bit in the "/oic/sec/pstat.tm" Property back to 0 (FALSE). If the Initiate Secure  
4485 Software Update bit of "/oic/sec/pstat.tm" is set to 0 (FALSE) by a Client, it has no effect on the  
4486 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  
4489 the 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  
4493 Device state(s) prior to completion of the update.

4494 The Device itself may indicate that it is autonomously initiating a software version check/update  
4495 or 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  
4520 be 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  
4523 document.
- 4524 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 4525 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of  
4526 the higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and  
4527 a later document version defines a security level 1, the Device could be evaluated and  
4528 classified as level 1 if it meets level 1 requirements).
- 4529 – The security levels may need to be visible to the end user.

4530 **14.8 Security Profiles**

4531 **14.8.1 Security Profiles General**

4532 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This  
4533 need for differentiation is based on the requirements from different verticals such as industrial  
4534 and health care and may extend into smart home. This differentiation is distinct from device  
4535 classification (e.g. IETF RFC 7228)

4536 These categories of security differentiation may include, but is not limited to:

4537 1) Security Hardening and assurances criteria

4538 2) Identity Attestation

4539 3) Certificate/Trust

4540 4) Onboarding Technique

4541 5) Regulatory Compliance

4542 a) Data at rest

4543 b) Data in transit

4544 6) Cipher Suites – Crypto Algorithms & Curves

4545 7) Key Length

4546 8) Secure Boot/Update

4547 Each Security Profile definition must specify the version or versions of the OCF Security  
4548 Specification(s) that form a baseline set of normative requirements. The profile definition may  
4549 include security requirements that supersede baseline requirements (not to relax security  
4550 requirements).

4551 Security Profiles have the following properties:

4552 – A given profile definition is not specific to the version of the document that defines it. For  
4553 example, the profile may remain constant for subsequent OCF Security Specification versions.

4554 – A specific OCF Device and platform combination may be used to satisfy the security profile.

4555 – Profiles may have overlapping criteria; hence it may be possible to satisfy multiple profiles  
4556 simultaneously.

4557 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found  
4558 to satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the  
4559 document, and a later document version defines a security profile Black, the device could be  
4560 evaluated and classified as profile Black if it meets profile Black requirements).

4561 – A machine-readable representation of compliance results specifically describing profiles  
4562 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or  
4563 manifest may contain security profiles attributes).

4564 **14.8.2 Identification of Security Profiles (Normative)**

4565 **14.8.2.1 Security Profiles in Prior Documents**

4566 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles  
4567 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in  
4568 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use  
4569 the OCF Security Specification version to characterize expected security behaviour.

4570 **14.8.2.2 Security Profile Resource Definition**

4571 The "oic.sec.sp" Resource is used by the OCF Device to show which OCF Security Profiles the  
4572 OCF Device is capable of supporting and which are authorized for use by the OCF Security

4573 Domain owner. Properties of the Resource identify which OCF Security Profile is currently  
 4574 operational. The ocfSecurityProfileOID value type shall represent OID values and may reference  
 4575 an entry in the form of strings (UTF-8).

4576 "oic.sec.sp" Resource is defined in Table 76.

4577 **Table 76 – 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

4578 Table 77 defines the Properties of "oic.sec.sp".

4579 **Table 77 – 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")

4580 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or  
 4581 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4582 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)  
 4583 private(4) enterprise(1) OCF(51414) }

4584  
 4585 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4586  
 4587 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4588  
 4589 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }  
 4590 --The Security Profile is not specified

4591 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }  
 4592 --This specifies the OCF Baseline Security Profile(s)

4593 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }  
 4594 --This specifies the OCF Black Security Profile(s)

4595 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }  
 4596 --This specified the OCF Blue Security Profile(s)

4597 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }  
 4598 --This specifies the OCF Purple Security Profile(s)

4599  
 4600 --versioned Security Profiles  
 4601 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)  
 4602 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4603 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}  
 4604 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4605 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}  
 4606 --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"

4607 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}  
 4608 --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"

4609 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}  
 4610 --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"

4611  
 4612 ocfSecurityProfileOID ::= UTF8String

4613

4614 **14.8.3 Security Profiles**

4615 **14.8.3.1 Security Profiles General**

4616 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to  
4617 the Security Profile clauses for additional details).

4618 The OCF Conformance criteria may require vendor attestation that establishes the expected  
4619 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific  
4620 requirements).

4621 **14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)**

4622 The Security Profile "sp-unspecified-v0" is reserved for future use.

4623 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4624 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where  
4625 the "/oic/sec/sp" Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the  
4626 "supportedprofiles" Property of the "/oic/sec/sp" Resource.

4627 It indicates the OCF Device satisfies the normative security requirements for this document.

4628 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-  
4629 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other  
4630 profiles.

4631 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to  
4632 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

4633 **14.8.3.4 Security Profile Black (sp-black-v0)**

4634 **14.8.3.4.1 Black Profile General**

4635 The need for Security Profile Black v0 is to support devices and manufacturers who wish to  
4636 certify their devices meeting this specific set of security criteria. A Device may satisfy the Black  
4637 requirements as well as requirements of other profiles, the Black Security Profile is not  
4638 necessarily mutually exclusive with other Security Profiles unless those requirements conflict with  
4639 the explicit requirements of the Black Security Profile.

4640 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4641 Security Profile Black devices could include any device a manufacturer wishes to certify at this  
4642 profile, but healthcare devices and industrial devices with additional security requirements are  
4643 the initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or  
4644 devices with exceptional profiles of trust bestowed upon them, may wish to certify at this profile;  
4645 these types of devices may include, but are not limited to the following:

4646 – Bridges (Mapping devices between ecosystems handling virtual devices from different  
4647 ecosystems)

4648 – Resource Directories (Devices trusted to manage OCF Security Domain resources)

4649 – Remote Access (Devices which have external access but can also act within the OCF  
4650 Security Domain)

4651 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)

4652 – Industrial Devices (Devices with advanced management, security and attestation  
4653 requirements)

4654 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4655 Every device with "currentprofile" Property of the "/oic/sec/sp" Resource designating a Security  
4656 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4657 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4658 – Support for AES 128 encryption for data at rest and in transit.
- 4659 – Hardening minimums: manufacturer assertion of secure credential storage
- 4660 – In 14) in enumerated item #10 "The "/oic/sec/cred" Resource should contain credential(s) if  
4661 required by the selected OTM" is changed to require the credential be stored: "The  
4662 "/oic/sec/cred" Resource shall contain credential(s)."
- 4663 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4664 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by  
4665 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4666 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-  
4667 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4668 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to  
4669 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4670 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework  
4671 described in the supporting documents:

- 4672 – Certificate Profile (See 9.4.2)
- 4673 – Certificate Policy (see Certificate Policy document: OCF-TSC-SWG-CP-D03-171101.docx)

4674 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4675 **14.8.3.5.1 Blue Profile General**

4676 The Security Profile Blue is used when manufacturers issue platform certificates for platforms  
4677 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is  
4678 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF  
4679 Security Domain owners evaluate manufacturer supplied certificates and attributed data to  
4680 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.  
4681 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may  
4682 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4683 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting  
4684 Criteria defined by OCF.

4685 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4686 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from  
4687 OCF Device vendor and where platform vendors may implement trusted platforms that may  
4688 conform to industry standards defining trusted platforms. The OCF Security Profile Blue specifies  
4689 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance  
4690 criteria produced by OCF conformance operations. The OCF Security Domain owner evaluates  
4691 these data when an OCF Device is onboarded into the OCF Security Domain. Based on this  
4692 evaluation the OCF Security Domain owner determines which Security Profile may be applied  
4693 during OCF Device operation. All OCF Device types may be considered for evaluation using the  
4694 OCF Security Profile Blue.

4695 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4696 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for  
4697 this document version are satisfied and the following additional criteria are satisfied.



4698 OCF Blue profile defines the following OCF Device quality assurances:

- 4699 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF  
4700 Device was hosted on one or more platforms that satisfies OCF Blue platform security  
4701 assurances and platform security and privacy functionality requirements.
- 4702 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF  
4703 and published by OCF in a machine readable format.
- 4704 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned  
4705 signing key.
- 4706 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its  
4707 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by  
4708 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4709 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in  
4710 its certificate.
- 4711 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF  
4712 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's  
4713 "securityprofiles" field.

4714 OCF Blue profile defines the following OCF Device security functionality:

- 4715 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage  
4716 functions are hardened by the platform.
- 4717 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials  
4718 using the "/oic/sec/cred" Resource where the "credusage" Property contains the value  
4719 "oic.sec.cred.mfgcert".
- 4720 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an  
4721 IEEE802.1AR IDevID and should verify the "TCG Endorsement Key Credential". All TCG-  
4722 defined manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the  
4723 "credusage" Property of the "/oic/sec/cred" Resource. They may be used in response to  
4724 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4725 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See  
4726 NIST SP 800-57).
- 4727 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST  
4728 SP 800-57).
- 4729 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST  
4730 SP 800-57).
- 4731 – OCF Device(s) should protect the "/oic/sec/cred" resource using the platform provided secure  
4732 storage.
- 4733 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned  
4734 certificates) using platform provided secure storage.
- 4735 – OCF Device(s) should check certificate revocation status for locally issued certificates.
- 4736 – OCF OBTs (aka DOTS) shall check certificate revocation status for all certificates in  
4737 manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation  
4738 fails and the connection is refused. The DOTS may disregard revocation status results if  
4739 unavailable.

4740 OCF Blue profile defines the following platform security assurances:

- 4741 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage  
4742 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL  
4743 Level 2.

4744 – Platforms implementing trusted platform functionality should be evaluated with a minimum  
4745 Common Criteria EAL Level 1.

4746 OCF Blue profile defines the following platform security and privacy functionality:

4747 – The Platform shall implement cryptographic service provider (CSP) functionality.

4748 – Platform CSP functionality shall include cryptographic algorithms, random number generation,  
4749 secure time.

4750 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST  
4751 SP 800-57).

4752 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See  
4753 NIST SP 800-57).

4754 – Platforms hosting OCF Device(s) should implement a platform identifier following  
4755 IEEE802.1AR or Trusted Computing Group(TCG) specifications.

4756 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF  
4757 Device(s) should supply TCG-defined manufacture certificates; also known as "TCG  
4758 Endorsement Key Credential" (which complies with IETF RFC 5280) and "TCG Platform  
4759 Credential" (which complies with IETF RFC 5755).

4760 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,  
4761 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

4762 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to  
4763 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

4764 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"  
4765 Property to one of the other values found in the "supportedprofiles" Property.

#### 4766 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**

4767 Every device with the "/oic/sec/sp" Resource designating "sp-purple-v0", as defined in clause  
4768 14.8.2 must support following minimum requirements

4769 – Hardening minimums: secure credential storage, software integrity validation, secure update.

4770 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension  
4771 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-  
4772 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"

4773 – The OCF Device shall include a X.509v3 OCFCLAttributes Extension (clause 9.4.2.2.7) in its  
4774 End-Entity Certificate when manufacturer certificate is used.

4775 Security Profile Purple has following optional security hardening requirements that the device can  
4776 additionally support.

4777 – Hardening additions: secure boot, hardware backed secure storage

4778 – The OCF Device shall include a X.509v3 OCFSecurityClaims Extension (clause 9.4.2.2.6) in  
4779 its End-Entity Certificate and it shall include corresponding OIDs to the hardening additions  
4780 implemented and attested by the vendor. If there is no additional support for hardening  
4781 requirements, X.509v3 OCFSecurityClaims Extension shall be omitted.

4782 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism  
4783 for security critical executables such as cryptographic modules or secure service applications,  
4784 and they should be validated before the execution. The key used for validating the integrity must  
4785 be pinned at the least to the validating software module.

4786 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4787 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM)  
4788 to be executed by the processor on power-on, and secure boot parameters to be provisioned by  
4789 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the  
4790 security critical executables and stop the boot process if any integrity of them is compromised.

4791 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile  
4792 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic  
4793 attacks.

4794 More details on security hardening guidelines for software integrity validation, secure boot,  
4795 secure update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4796 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA  
4797 Vetting Criteria defined by OCF.

4798 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-  
4799 purple-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other  
4800 profiles.

4801 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to  
4802 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

## 4803 **15 Device Type Specific Requirements**

### 4804 **15.1 Bridging Security**

#### 4805 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4806 The Bridge shall go through OCF ownership transfer as any other onboarder would.

4807 The software of an Bridge shall be field updatable. (This requirement need not be tested but can  
4808 be certified via a vendor declaration.)

4809 Each VOD shall be onboarded by an OCF OBT. Each Virtual Bridged Device should be  
4810 provisioned as appropriate in the Bridged Protocol. In other words, VODs and Virtual Bridged  
4811 Devices are treated the same way as physical Devices. They are entities that have to be  
4812 provisioned in their network.

4813 Each VOD shall implement the behaviour required by ISO/IEC 30118-1:2018 and this document.  
4814 Each VOD shall perform authentication, access control, and encryption according to the security  
4815 settings it received from the OCF OBT. Each Virtual Bridged Device shall implement the security  
4816 requirements of the Bridged Protocol.

4817 In addition, in order to be considered secure from an OCF perspective, the Bridge Platform shall  
4818 use appropriate ecosystem-specific security options for communication between the Virtual  
4819 Bridged Devices instantiated by the Bridge and Bridged Devices. This security shall include  
4820 mutual authentication, and encryption and integrity protection of messages in the bridged  
4821 ecosystem.

4822 A VOD may authenticate itself to the DOTS using the Manufacturer Certificate Based OTM (see  
4823 clause 7.3.6) with the Manufacturer Certificate and corresponding private key of the Bridge which  
4824 instantiated that VOD.

4825 A VOD may authenticate itself to the OCF Cloud (see clause 10.5.2) using the Manufacturer  
4826 Certificate and corresponding private key of the Bridge which instantiated that VOD.

#### 4827 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

##### 4828 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4829 For AllJoyn translator, an OCF OBT shall be able to block the communication of all OCF Devices  
4830 with all Bridged Devices that don't communicate securely with the Bridge, by using the Bridge  
4831 Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

##### 4832 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4833 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4834 communicate securely with the Bridge.

##### 4835 **15.1.2.3 Additional Security Requirements specific to the oneM2M Protocols**

4836 The Bridge shall implement oneM2M application access control as defined in the oneM2M  
4837 Release 3 Specifications.

4838 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4839 communicate securely with the Bridge.

##### 4840 **15.1.2.4 Additional Security Requirements specific to the U+ Protocol**

4841 A Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4842 communicate securely with the Bridge.

4843 **15.1.2.5 Additional Security Requirements specific to the Z-Wave Protocol**

4844 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4845 communicate securely with the Bridge.

4846 **15.1.2.6 Additional Security Requirements specific to the Zigbee Protocol**

4847 An Bridge shall block the communication of all OCF Devices with all Bridged Devices that don't  
4848 communicate securely with the Bridge.

4849

4850

4851

4852

4853

4854

4855

4856

4857

4858

4859

4860

4861

4862

4863

4864

4865

4866

4867

4868

4869

4870 .

4871 **Annex A**  
4872 **(informative)**  
4873 **Access Control Examples**

4874 **Example OCF ACL Resource**

4875 Figure A-1 shows how a "/oic/sec/acl2" Resource could be configured to enforce an example  
4876 access policy on the Server.

```
4877 {  
4878   "aclist2": [  
4879     {  
4880       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve,  
4881       Update, Delete and Notify)  
4882       "subject": {"uuid": "XXXX-...-XX01"},  
4883       "resources": [  
4884         {"href": "/oic/sh/light/1"},  
4885         {"href": "/oic/sh/temp/0"}  
4886       ],  
4887       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC  
4888       "validity": [  
4889         // The period starting at 18:00:00 UTC, on January 1, 2015 and  
4890         // ending at 07:00:00 UTC on January 2, 2015  
4891         "period": ["20150101T180000Z/20150102T070000Z"],  
4892         // Repeats the {period} every week until the last day of Jan. 2015.  
4893         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]  
4894       ],  
4895       "aceid": 1  
4896     }  
4897   ],  
4898   // An ACL provisioning and management service should be identified as  
4899   // the resource owner  
4900   "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"  
4901 }
```

4902 **Figure A-1 – Example "/oic/sec/acl2" Resource**

4903 **Example AMS**

4904 Figure A-2 demonstrates how the "/oic/sec/amacl" Resource should be configured to achieve this  
4905 objective.

```
4906 {  
4907   "resources": [  
4908     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was  
4909     // supplied then use the sacl validation credential to enforce access.  
4910     {"href": "/oic/sh/light/1"},  
4911     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was  
4912     // supplied then use the sacl validation credential to enforce access.  
4913     {"href": "/oma/3"},
```

```
4914 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4915 // the sac1 validation credential to enforce access.
4916 {"wc": ""}]
4917 }
4918
```

**Figure A-2 Example "/oic/sec/amacl" Resource**

4919  
4920  
4921

## Annex B (Informative) Execution Environment Security Profiles

4922 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security  
4923 robustness requirements meeting all IOT applications and services will not serve the needs of  
4924 OCF, and security profiles of varying degree of robustness (trustworthiness), cost and complexity  
4925 have to be defined. To address a large ecosystem of vendors, the profiles can only be defined as  
4926 requirements and the exact solutions meeting those requirements are specific to the vendors'  
4927 open or proprietary implementations, and thus in most part outside scope of this document.

4928 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228  
4929 (Terminology for constrained node networks) methodology, we limit the number of security  
4930 profiles to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities  
4931 criteria for each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4932 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are  
4933 either capable of no security functionality or easily breakable security that depend on  
4934 environmental (e.g. availability of human) factors to perform security functions. This means the  
4935 class 0 will not be equipped with an SEE.

4936

**Table B.1 – OCF Security Profile**

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4937 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of  
4938 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,  
4939 memory, storage).



4940  
4941  
4942

## Annex C (normative) Resource Type definitions

### 4943 C.1 List of Resource Type definitions

4944 Table C.1 contains the list of defined security resources in this document.

4945 **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

4946

### 4947 C.2 Account Token

#### 4948 C.2.1 Introduction

4949 Sign-up using generic account provider.

#### 4950 C.2.2 Well-known URI

4951 /oic/sec/account

#### 4952 C.2.3 Resource type

4953 The Resource Type is defined as: "oic.r.account".

#### 4954 C.2.4 OpenAPI 2.0 definition

```
4955 {  
4956   "swagger": "2.0",  
4957   "info": {  
4958     "title": "Account Token",  
4959     "version": "20190111",  
4960     "license": {  
4961       "name": "OCF Data Model License",  
4962       "url":  
4963         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
4964         CENSE.md",  
4965         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
4966         reserved."  
4967     }  
4968     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
```

```

4969     },
4970     "schemes": ["http"],
4971     "consumes": ["application/json"],
4972     "produces": ["application/json"],
4973     "paths": {
4974         "/oic/sec/account" : {
4975             "post": {
4976                 "description": "Sign-up using generic account provider.\n",
4977                 "parameters": [
4978                     { "$ref": "#/parameters/interface" },
4979                     {
4980                         "name": "body",
4981                         "in": "body",
4982                         "required": true,
4983                         "schema": { "$ref": "#/definitions/Account-request" },
4984                         "x-example":
4985                             {
4986                                 "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
4987                                 "authprovider" : "github",
4988                                 "accesstoken" : "8802f2eaf8b5e147a936"
4989                             }
4990                     }
4991                 ],
4992                 "responses": {
4993                     "204": {
4994                         "description": "2.04 Changed respond with required and optional information\n",
4995                         "x-example":
4996                             {
4997                                 "rt": ["oic.r.account"],
4998                                 "accesstoken" : "0f3d9f7fe5491d54077d",
4999                                 "refreshtoken" : "00fe4644a6fbe5324eec",
5000                                 "expiresin" : 3600,
5001                                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
5002                                 "redirecturi" : "coaps+tcp://example.com:443"
5003                             },
5004                         "schema": { "$ref": "#/definitions/Account-response" }
5005                     }
5006                 }
5007             },
5008             "delete": {
5009                 "description": "Delete a device. This also removes all resources in the device on cloud
5010 side.\nexample: /oic/account?di=9cfbeb8e-5ale-4dlc-9d01-
5011 00c04fd430c8&accesstoken=0f3d9f7fe5491d54077d\n",
5012                 "parameters": [
5013                     { "$ref": "#/parameters/interface" }
5014                 ],
5015                 "responses": {
5016                     "202": {
5017                         "description": "2.02 Deleted response informing the device is successfully
5018 deleted.\n"
5019                     }
5020                 }
5021             }
5022         },
5023     },
5024     "parameters": {
5025         "interface" : {
5026             "in" : "query",
5027             "name" : "if",
5028             "type" : "string",
5029             "enum" : ["oic.if.baseline"]
5030         }
5031     },
5032     "definitions": {
5033         "Account-request" : {
5034             "properties": {
5035                 "authprovider": {
5036                     "description": "The name of Authorization Provider through which Access Token was
5037 obtained",
5038                     "type": "string"
5039                 }
5040             }
5041         }
5042     }
5043 }

```

```

5040     "accesstoken" : {
5041         "description": "Access-Token used for communication with OCF Cloud after account creation",
5042         "pattern": "(?!$|\\s+).*",
5043         "type": "string"
5044     },
5045     "di": {
5046         "description": "Format pattern according to IETF RFC 4122.",
5047         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5048         "type": "string"
5049     }
5050 },
5051 "type" : "object",
5052 "required": ["di", "accesstoken"]
5053 },
5054 "Account-response": {
5055     "properties": {
5056         "expiresin": {
5057             "description": "Access-Token remaining life time in seconds (-1 if permanent)",
5058             "readOnly": true,
5059             "type": "integer"
5060         },
5061         "rt": {
5062             "description": "Resource Type of the Resource",
5063             "items": {
5064                 "maxLength": 64,
5065                 "type": "string",
5066                 "enum" : ["oic.r.account"]
5067             },
5068             "minItems": 1,
5069             "maxItems": 1,
5070             "readOnly": true,
5071             "type": "array"
5072         },
5073         "refreshToken" : {
5074             "description": "Refresh token can be used to refresh the Access Token before getting
5075 expired",
5076             "pattern": "(?!$|\\s+).*",
5077             "readOnly": true,
5078             "type": "string"
5079         },
5080         "uid" : {
5081             "description": "Format pattern according to IETF RFC 4122.",
5082             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5083             "type": "string"
5084         },
5085         "accesstoken" : {
5086             "description": "Access-Token used for communication with cloud after account creation",
5087             "pattern": "(?!$|\\s+).*",
5088             "type": "string"
5089         },
5090         "n": {
5091             "$ref":
5092 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5093 schema.json#/definitions/n"
5094         },
5095         "id": {
5096             "$ref":
5097 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5098 schema.json#/definitions/id"
5099         },
5100         "redirecturi" : {
5101             "description": "Using this URI, the Client needs to reconnect to a redirected OCF Cloud.
5102 If provided, this value shall be used by the Device instead of Mediator-provided URI during the
5103 Device Registration.",
5104             "readOnly": true,
5105             "type": "string"
5106         },
5107         "if": {
5108             "description": "The interface set supported by this resource",
5109             "items": {
5110                 "enum": [

```

```

5111         "oic.if.baseline"
5112     ],
5113     "type": "string"
5114 },
5115     "minItems": 1,
5116     "maxItems": 1,
5117     "uniqueItems": true,
5118     "readOnly": true,
5119     "type": "array"
5120 }
5121 },
5122     "type" : "object",
5123     "required": ["accesstoken", "refreshtoken", "expiresin", "uid"]
5124 }
5125 }
5126 }
5127

```

### 5128 C.2.5 Property definition

5129 Table C.2 defines the Properties that are part of the "oic.r.account" Resource Type.

5130 **Table C.2 – The Property definitions of the Resource with type "rt" = "oic.r.account".**

Property name	Value type	Mandatory	Access mode	Description
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
authprovider	string	No	Read Write	The name of Authorization Provider through which Access Token was obtained
accesstoken	string	Yes	Read Write	Access-Token used for communication with OCF Cloud after account creation
id	multiple types: see schema	No	Read Write	
refreshtoken	string	Yes	Read Only	Refresh token can be used to refresh the Access Token before getting expired
rt	array: see schema	No	Read Only	Resource Type of the Resource
accesstoken	string	Yes	Read Write	Access-Token used for communication with cloud after account creation
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds

				(-1 if permanent)
if	array: schema	see	No	Read Only
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.
n	multiple types: see schema		No	Read Write

5131 **C.2.6 CRUDN behaviour**

5132 Table C.3 defines the CRUDN operations that are supported on the "oic.r.account" Resource  
5133 Type.

5134 **Table C.3 – The CRUDN operations of the Resource with type "rt" = "oic.r.account".**

Create	Read	Update	Delete	Notify
		post	delete	

5135 **C.3 Access Control List**

5136 **C.3.1 Introduction**

5137 This Resource specifies the local access control list.  
5138 When used without query parameters, all the ACE entries are returned.  
5139 When used with a subjectuid, only the ACEs with the specified  
5140 subjectuid are returned. If subjectuid and Resources are specified,  
5141 only the ACEs with the specified subjectuid and Resource hrefs are  
5142 returned.

5143 **C.3.2 Well-known URI**

5144 /oic/sec/acl

5145 **C.3.3 Resource type**

5146 The Resource Type is defined as: "oic.r.acl".

5147 **C.3.4 OpenAPI 2.0 definition**

```
5148 {
5149   "swagger": "2.0",
5150   "info": {
5151     "title": "Access Control List",
5152     "version": "v1.1-20161213",
5153     "license": {
5154       "name": "OCF Data Model License",
5155       "url":
5156         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5157         CENSE.md",
5158       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5159       reserved."

```

```

5160     },
5161     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5162   },
5163   "schemes": ["http"],
5164   "consumes": ["application/json"],
5165   "produces": ["application/json"],
5166   "paths": {
5167     "/oic/sec/acl" : {
5168       "get": {
5169         "description": "This Resource specifies the local access control list.\nWhen used without
5170 query parameters, all the ACE entries are returned.\nWhen used with a subjectuuid, only the ACEs
5171 with the specified\nsubjectuuid are returned. If subjectuuid and Resources are specified,\nonly the
5172 ACEs with the specified subjectuuid and Resource hrefs are\nreturned.\n",
5173         "parameters": [
5174           { "$ref": "#/parameters/interface" },
5175           { "$ref": "#/parameters/ace-filtered-uuid" },
5176           { "$ref": "#/parameters/ace-filtered-resources" }
5177         ],
5178         "responses": {
5179           "200": {
5180             "description": "",
5181             "x-example":
5182               {
5183                 "rt": ["oic.r.acl"],
5184                 "aclist": {
5185                   "aces": [
5186                     {
5187                       "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5188                       "resources": [
5189                         {
5190                           "href": "coaps://IP-ADDR/temp",
5191                           "rel": "some-rel",
5192                           "rt": ["oic.r.temperature"],
5193                           "if": ["oic.if.a"]
5194                         },
5195                         {
5196                           "href": "coaps://IP-ADDR/temp",
5197                           "rel": "some-rel",
5198                           "rt": ["oic.r.temperature"],
5199                           "if": ["oic.if.s"]
5200                         }
5201                       ],
5202                       "permission": 31,
5203                       "validity": [
5204                         {
5205                           "period": "20160101T180000Z/20170102T070000Z",
5206                           "recurrence": [ "DSTART:XXXXX",
5207 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5208                         },
5209                         {
5210                           "period": "20160101T180000Z/PT5H30M",
5211                           "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5212                         }
5213                       ]
5214                     }
5215                   ]
5216                 },
5217                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5218               },
5219               "schema": { "$ref": "#/definitions/Acl" }
5220             },
5221             "400": {
5222               "description": "The request is invalid."
5223             }
5224           }
5225         },
5226         "post": {
5227           "description": "Updates the ACL Resource with the provided values. ACEs provided\nin the
5228 update not currently in the ACL are added. ACEs that already\nexist in the ACL are ignored.\n\nNote
5229 that for the purposes of update, equivalency is determined\nby comparing the ACE subjectuuid,
5230 permission, string comparisons\nof all validity elements, and string comparisons of all

```

```

5231 Resource\nhrefs.\n",
5232     "parameters": [
5233         {"$ref": "#/parameters/interface"},
5234         {
5235             "name": "body",
5236             "in": "body",
5237             "required": true,
5238             "schema": { "$ref": "#/definitions/Acl" },
5239             "x-example":
5240                 {
5241                     "acllist": {
5242                         "aces": [
5243                             {
5244                                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5245                                 "resources": [
5246                                     {
5247                                         "href": "coaps://IP-ADDR/temp",
5248                                         "rel": "some-rel",
5249                                         "rt": ["oic.r.temperature"],
5250                                         "if": ["oic.if.a"]
5251                                     },
5252                                     {
5253                                         "href": "coaps://IP-ADDR/temp",
5254                                         "rel": "some-rel",
5255                                         "rt": ["oic.r.temperature"],
5256                                         "if": ["oic.if.s"]
5257                                     }
5258                                 ],
5259                                 "permission": 31,
5260                                 "validity": [
5261                                     {
5262                                         "period": "20160101T180000Z/20170102T070000Z",
5263                                         "recurrence": [ "DSTART:XXXXX",
5264 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5265                                     },
5266                                     {
5267                                         "period": "20160101T180000Z/PT5H30M",
5268                                         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5269                                     }
5270                                 ]
5271                             }
5272                         ],
5273                     },
5274                     "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5275                 }
5276             }
5277         ],
5278         "responses": {
5279             "400": {
5280                 "description": "The request is invalid."
5281             },
5282             "201": {
5283                 "description": "The ACL entry/entries is/are created."
5284             },
5285             "204": {
5286                 "description": "The ACL entry/entries is/are updated."
5287             }
5288         }
5289     },
5290     "delete": {
5291         "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5292 ACE entries are deleted.\nWhen DELETE is used with a subjectuuid, only the ACEs with the
5293 specified\nsubjectuuid are deleted. If subjectuuid and Resources are specified,\nonly the ACEs with
5294 the specified subjectuuid and Resource hrefs are\ndeleted.\n",
5295         "parameters": [
5296             {"$ref": "#/parameters/interface"},
5297             {"$ref": "#/parameters/ace-filtered-uuid"},
5298             {"$ref": "#/parameters/ace-filtered-resources"}
5299         ],
5300         "responses": {
5301             "200": {

```

```

5302         "description" : "The matching ACEs or the entire ACL Resource has been successfully
5303 deleted."
5304     },
5305     "400": {
5306         "description" : "The request is invalid."
5307     }
5308 }
5309 }
5310 }
5311 },
5312 "parameters": {
5313     "interface" : {
5314         "in" : "query",
5315         "name" : "if",
5316         "type" : "string",
5317         "enum" : ["oic.if.baseline"]
5318     },
5319     "ace-filtered-uuid" : {
5320         "in" : "query",
5321         "name" : "subjectuuid",
5322         "required" : false,
5323         "type" : "string",
5324         "description" : "Only applies to ACEs with the specified subject UUID.",
5325         "x-example" : "se61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5326     },
5327     "ace-filtered-resources" : {
5328         "in" : "query",
5329         "name" : "resources",
5330         "required" : false,
5331         "type" : "string",
5332         "description" : "Only applies to ACEs with the specified subhjectuuid | and Resources href.",
5333         "x-example" : "coaps://IP-ADDR/temp"
5334     }
5335 },
5336 "definitions": {
5337     "Acl" : {
5338         "properties": {
5339             "owneruuid": {
5340                 "description": "The value identifies the unique Resource owner\nFormat pattern according
5341 to IETF RFC 4122.",
5342                 "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}$",
5343                 "type": "string"
5344             },
5345             "rt": {
5346                 "description": "Resource Type of the Resource.",
5347                 "items": {
5348                     "maxLength": 64,
5349                     "type": "string",
5350                     "enum": ["oic.r.acl"]
5351                 },
5352                 "minItems": 1,
5353                 "readOnly": true,
5354                 "type": "array"
5355             },
5356             "aclist": {
5357                 "description": "Subject-based Access Control Entries in the ACL Resource.",
5358                 "properties": {
5359                     "aces": {
5360                         "items": {
5361                             "properties": {
5362                                 "permission": {
5363                                     "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask
5364 indicating permissions.",
5365                                     "x-detail-desc": [
5366                                         "0 - No permissions.",
5367                                         "1 - Create permission is granted.",
5368                                         "2 - Read, observe, discover permission is granted.",
5369                                         "4 - Write, update permission is granted.",
5370                                         "8 - Delete permission is granted.",
5371                                         "16 - Notify permission is granted."
5372                                     ],

```



```

5373         "maximum": 31,
5374         "minimum": 0,
5375         "type": "integer"
5376     },
5377     "resources": {
5378         "description": "References the application's Resources to which a security
5379 policy applies.",
5380         "items": {
5381             "properties": {
5382                 "anchor": {
5383                     "description": "This is used to override the context URI e.g. override the
5384 URI of the containing collection.",
5385                     "format": "uri",
5386                     "maxLength": 256,
5387                     "type": "string"
5388                 },
5389                 "di": {
5390                     "description": "The Device ID\nFormat pattern according to IETF RFC 4122.",
5391                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
5392 [a-fA-F0-9]{12}$",
5393                     "type": "string"
5394                 },
5395                 "eps": {
5396                     "description": "the Endpoint information of the target Resource.",
5397                     "items": {
5398                         "properties": {
5399                             "ep": {
5400                                 "description": "Transport Protocol Suite + Endpoint Locator.",
5401                                 "format": "uri",
5402                                 "type": "string"
5403                             },
5404                             "pri": {
5405                                 "description": "The priority among multiple Endpoints.",
5406                                 "minimum": 1,
5407                                 "type": "integer"
5408                             }
5409                         },
5410                         "type": "object"
5411                     },
5412                     "type": "array"
5413                 },
5414                 "href": {
5415                     "description": "This is the target URI, it can be specified as a Relative
5416 Reference or fully-qualified URI.",
5417                     "format": "uri",
5418                     "maxLength": 256,
5419                     "type": "string"
5420                 },
5421                 "if": {
5422                     "description": "The interface set supported by this Resource.",
5423                     "items": {
5424                         "enum": [
5425                             "oic.if.baseline",
5426                             "oic.if.ll",
5427                             "oic.if.b",
5428                             "oic.if.rw",
5429                             "oic.if.r",
5430                             "oic.if.a",
5431                             "oic.if.s"
5432                         ],
5433                         "type": "string"
5434                     },
5435                     "minItems": 1,
5436                     "type": "array"
5437                 },
5438                 "ins": {
5439                     "description": "The instance identifier for this web link in an array of
5440 web links - used in collections.",
5441                     "type": "integer"
5442                 },
5443                 "p": {

```

```

5444         "description": "Specifies the framework policies on the Resource
5445 referenced by the target URI.",
5446         "properties": {
5447             "bm": {
5448                 "description": "Specifies the framework policies on the Resource
5449 referenced by the target URI for e.g. observable and discoverable.",
5450                 "type": "integer"
5451             }
5452         },
5453         "required": [
5454             "bm"
5455         ],
5456         "type": "object"
5457     },
5458     "rel": {
5459         "description": "The relation of the target URI referenced by the link to
5460 the context URI.",
5461         "oneOf": [
5462             {
5463                 "default": [
5464                     "hosts"
5465                 ],
5466                 "items": {
5467                     "maxLength": 64,
5468                     "type": "string"
5469                 },
5470                 "minItems": 1,
5471                 "type": "array"
5472             },
5473             {
5474                 "default": "hosts",
5475                 "maxLength": 64,
5476                 "type": "string"
5477             }
5478         ]
5479     },
5480     "rt": {
5481         "description": "Resource Type of the Resource.",
5482         "items": {
5483             "maxLength": 64,
5484             "type": "string"
5485         },
5486         "minItems": 1,
5487         "type": "array"
5488     },
5489     "title": {
5490         "description": "A title for the link relation. Can be used by the UI to
5491 provide a context.",
5492         "maxLength": 64,
5493         "type": "string"
5494     },
5495     "type": {
5496         "default": "application/cbor",
5497         "description": "A hint at the representation of the Resource referenced by
5498 the target URI. This represents the media types that are used for both accepting and emitting.",
5499         "items": {
5500             "maxLength": 64,
5501             "type": "string"
5502         },
5503         "minItems": 1,
5504         "type": "array"
5505     }
5506 },
5507 "required": [
5508     "href",
5509     "rt",
5510     "if"
5511 ],
5512 "type": "object"
5513 },
5514 "type": "array"

```

```

5515         },
5516         "subjectuuid": {
5517             "anyOf": [
5518                 {
5519                     "description": "The id of the Device to which the ace applies to or \"*\
5520 for anonymous access.\",
5521                     "pattern": "^\\*$",
5522                     "type": "string"
5523                 },
5524                 {
5525                     "description": "Format pattern according to IETF RFC 4122.",
5526                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5527 fA-F0-9]{12}$",
5528                     "type": "string"
5529                 }
5530             ]
5531         },
5532         "validity": {
5533             "description": "validity is an array of time-pattern objects.",
5534             "items": {
5535                 "description": "The time-pattern contains a period and recurrence expressed in
5536 RFC5545 syntax.",
5537                 "properties": {
5538                     "period": {
5539                         "description": "String represents a period using the RFC5545 Period.",
5540                         "type": "string"
5541                     },
5542                     "recurrence": {
5543                         "description": "String array represents a recurrence rule using the
5544 RFC5545 Recurrence.",
5545                         "items": {
5546                             "type": "string"
5547                         },
5548                         "type": "array"
5549                     }
5550                 },
5551                 "required": [
5552                     "period"
5553                 ],
5554                 "type": "object"
5555             },
5556             "type": "array"
5557         }
5558     },
5559     "required": [
5560         "resources",
5561         "permission",
5562         "subjectuuid"
5563     ],
5564     "type": "object"
5565 },
5566 "type": "array"
5567 }
5568 },
5569 "required": [
5570     "aces"
5571 ],
5572 "type": "object"
5573 },
5574 "n": {
5575     "$ref":
5576 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5577 schema.json#/definitions/n"
5578 },
5579 "id": {
5580     "$ref":
5581 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5582 schema.json#/definitions/id"
5583 },
5584 "if": {
5585     "description": "The interface set supported by this Resource.",

```

```

5586     "items": {
5587         "enum": [
5588             "oic.if.baseline"
5589         ],
5590         "type": "string"
5591     },
5592     "minItems": 1,
5593     "readOnly": true,
5594     "type": "array"
5595 }
5596 },
5597 "type" : "object",
5598 "required": ["aclist", "rowneruuid"]
5599 }
5600 }
5601 }
5602

```

5603 **C.3.5 Property definition**

5604 Table C.4 defines the Properties that are part of the "oic.r.acl" Resource Type.

5605 **Table C.4 – The Property definitions of the Resource with type "rt" = "oic.r.acl".**

Property name	Value type	Mandatory	Access mode	Description
aclist	object: see schema	Yes	Read Write	Subject-based Access Control Entries in the ACL Resource.
rowneruuid	string	Yes	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
id	multiple types: see schema	No	Read Write	

5606 **C.3.6 CRUDN behaviour**

5607 Table C.5 defines the CRUDN operations that are supported on the "oic.r.acl" Resource Type.

5608 **Table C.5 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5609 **C.4 Access Control List-2**

5610 **C.4.1 Introduction**

5611 This Resource specifies the local access control list.

5612 When used without query parameters, all the ACE entries are returned.

5613 When used with a query parameter, only the ACEs matching the specified

5614 parameter are returned.

5615

## 5616 C.4.2 Well-known URI

5617 /oic/sec/acl2

## 5618 C.4.3 Resource type

5619 The Resource Type is defined as: "oic.r.acl2".

## 5620 C.4.4 OpenAPI 2.0 definition

```
5621 {
5622   "swagger": "2.0",
5623   "info": {
5624     "title": "Access Control List-2",
5625     "version": "20190111",
5626     "license": {
5627       "name": "OCF Data Model License",
5628       "url":
5629 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5630 CENSE.md",
5631       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5632 reserved."
5633     },
5634     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5635   },
5636   "schemes": ["http"],
5637   "consumes": ["application/json"],
5638   "produces": ["application/json"],
5639   "paths": {
5640     "/oic/sec/acl2" : {
5641       "get": {
5642         "description": "This Resource specifies the local access control list.\nWhen used without
5643 query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs
5644 matching the specified\nparameter are returned.\n",
5645         "parameters": [
5646           {"$ref": "#/parameters/interface"},
5647           {"$ref": "#/parameters/ace-filtered"}
5648         ],
5649         "responses": {
5650           "200": {
5651             "description": "",
5652             "x-example":
5653             {
5654               "rt" : ["oic.r.acl2"],
5655               "aclist2": [
5656                 {
5657                   "aceid": 1,
5658                   "subject": {
5659                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5660                     "role": "SOME_STRING"
5661                   },
5662                   "resources": [
5663                     {
5664                       "href": "/light",
5665                       "rt": ["oic.r.light"],
5666                       "if": ["oic.if.baseline", "oic.if.a"]
5667                     },
5668                     {
5669                       "href": "/door",
5670                       "rt": ["oic.r.door"],
5671                       "if": ["oic.if.baseline", "oic.if.a"]
5672                     }
5673                   ],
5674                   "permission": 24
5675                 },
5676               {
5677                 "aceid": 2,
5678                 "subject": {
5679                   "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
```

```

5680     },
5681     "resources": [
5682     {
5683         "href": "/light",
5684         "rt": ["oic.r.light"],
5685         "if": ["oic.if.baseline", "oic.if.a"]
5686     },
5687     {
5688         "href": "/door",
5689         "rt": ["oic.r.door"],
5690         "if": ["oic.if.baseline", "oic.if.a"]
5691     }
5692 ],
5693 "permission": 24
5694 },
5695 {
5696     "aceid": 3,
5697     "subject": {"conntype": "anon-clear"},
5698     "resources": [
5699     {
5700         "href": "/light",
5701         "rt": ["oic.r.light"],
5702         "if": ["oic.if.baseline", "oic.if.a"]
5703     },
5704     {
5705         "href": "/door",
5706         "rt": ["oic.r.door"],
5707         "if": ["oic.if.baseline", "oic.if.a"]
5708     }
5709 ],
5710 "permission": 16,
5711 "validity": [
5712     {
5713         "period": "20160101T180000Z/20170102T070000Z",
5714         "recurrence": [ "DSTART:XXXXX",
5715 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5716     },
5717     {
5718         "period": "20160101T180000Z/PT5H30M",
5719         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5720     }
5721 ]
5722 }
5723 ],
5724 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5725 },
5726 "schema": { "$ref": "#/definitions/Acl2" }
5727 },
5728 "400": {
5729     "description": "The request is invalid."
5730 }
5731 }
5732 },
5733 "post": {
5734     "description": "Updates the ACL Resource with the provided ACEs.\n\nACEs provided in the
5735 update with aceids not currently in the ACL\nResource are added.\n\nACEs provided in the update with
5736 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL Resource.\n\nACEs provided in
5737 the update without aceid properties are added and\nassigned unique aceids in the ACL Resource.\n",
5738     "parameters": [
5739         {"$ref": "#/parameters/interface"},
5740         {"$ref": "#/parameters/ace-filtered"},
5741     ],
5742     "name": "body",
5743     "in": "body",
5744     "required": true,
5745     "schema": { "$ref": "#/definitions/Acl2-Update" },
5746     "x-example":
5747     {
5748         "aclist2": [
5749             {
5750                 "aceid": 1,

```

```

5751         "subject": {
5752             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5753             "role": "SOME_STRING"
5754         },
5755         "resources": [
5756             {
5757                 "href": "/light",
5758                 "rt": ["oic.r.light"],
5759                 "if": ["oic.if.baseline", "oic.if.a"]
5760             },
5761             {
5762                 "href": "/door",
5763                 "rt": ["oic.r.door"],
5764                 "if": ["oic.if.baseline", "oic.if.a"]
5765             }
5766         ],
5767         "permission": 24
5768     },
5769     {
5770         "aceid": 3,
5771         "subject": {
5772             "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5773         },
5774         "resources": [
5775             {
5776                 "href": "/light",
5777                 "rt": ["oic.r.light"],
5778                 "if": ["oic.if.baseline", "oic.if.a"]
5779             },
5780             {
5781                 "href": "/door",
5782                 "rt": ["oic.r.door"],
5783                 "if": ["oic.if.baseline", "oic.if.a"]
5784             }
5785         ],
5786         "permission": 24
5787     }
5788 ],
5789     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5790 }
5791 }
5792 ],
5793 "responses": {
5794     "400": {
5795         "description": "The request is invalid."
5796     },
5797     "201": {
5798         "description": "The ACL entry is created."
5799     },
5800     "204": {
5801         "description": "The ACL entry is updated."
5802     }
5803 }
5804 },
5805 "delete": {
5806     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5807 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
5808 the\nspecified parameter are deleted.\n",
5809     "parameters": [
5810         {"$ref": "#/parameters/interface"},
5811         {"$ref": "#/parameters/ace-filtered"}
5812     ],
5813     "responses": {
5814         "200": {
5815             "description": "The matching ACEs or the entire ACL Resource has been successfully
5816 deleted."
5817         },
5818         "400": {
5819             "description": "The request is invalid."
5820         }
5821     }

```

```

5822     }
5823   },
5824 },
5825 "parameters": {
5826   "interface" : {
5827     "in" : "query",
5828     "name" : "if",
5829     "type" : "string",
5830     "enum" : ["oic.if.baseline"]
5831   },
5832   "ace-filtered" : {
5833     "in" : "query",
5834     "name" : "aceid",
5835     "required" : false,
5836     "type" : "integer",
5837     "description" : "Only applies to the ACE with the specified aceid.",
5838     "x-example" : 2112
5839   }
5840 },
5841 "definitions": {
5842   "Acl2" : {
5843     "properties": {
5844       "rowneruuid" : {
5845         "description": "The value identifies the unique Resource owner\nFormat pattern according
5846 to IETF RFC 4122.",
5847         "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}$",
5848         "type": "string"
5849       },
5850       "rt" : {
5851         "description": "Resource Type of the Resource.",
5852         "items": {
5853           "maxLength": 64,
5854           "type": "string",
5855           "enum": ["oic.r.acl2"]
5856         },
5857         "minItems": 1,
5858         "maxItems": 1,
5859         "readOnly": true,
5860         "type": "array"
5861       },
5862       "aclist2" : {
5863         "description": "Access Control Entries in the ACL Resource.",
5864         "items": {
5865           "properties": {
5866             "aceid": {
5867               "description": "An identifier for the ACE that is unique within the ACL. In cases
5868 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5869               "minimum": 1,
5870               "type": "integer"
5871             },
5872             "permission": {
5873               "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5874 permissions.",
5875               "x-detail-desc": [
5876                 "0 - No permissions",
5877                 "1 - Create permission is granted",
5878                 "2 - Read, observe, discover permission is granted",
5879                 "4 - Write, update permission is granted",
5880                 "8 - Delete permission is granted",
5881                 "16 - Notify permission is granted"
5882               ],
5883               "maximum": 31,
5884               "minimum": 0,
5885               "type": "integer"
5886             },
5887             "resources": {
5888               "description": "References the application's Resources to which a security policy
5889 applies.",
5890               "items": {
5891                 "description": "Each Resource must have at least one of these properties set.",
5892                 "properties": {

```



```

5893         "href": {
5894             "description": "When present, the ACE only applies when the href matches\nThis
5895 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5896             "format": "uri",
5897             "maxLength": 256,
5898             "type": "string"
5899         },
5900         "if": {
5901             "description": "When present, the ACE only applies when the if (interface)
5902 matches\nThe interface set supported by this Resource.",
5903             "items": {
5904                 "enum": [
5905                     "oic.if.baseline",
5906                     "oic.if.ll",
5907                     "oic.if.b",
5908                     "oic.if.rw",
5909                     "oic.if.r",
5910                     "oic.if.a",
5911                     "oic.if.s"
5912                 ],
5913                 "type": "string"
5914             },
5915             "minItems": 1,
5916             "type": "array"
5917         },
5918         "rt": {
5919             "description": "When present, the ACE only applies when the rt (Resource type)
5920 matches\nResource Type of the Resource.",
5921             "items": {
5922                 "maxLength": 64,
5923                 "type": "string"
5924             },
5925             "minItems": 1,
5926             "type": "array"
5927         },
5928         "wc": {
5929             "description": "A wildcard matching policy.",
5930             "pattern": "^[+*]$",
5931             "type": "string"
5932         }
5933     },
5934     "type": "object"
5935 },
5936 "type": "array"
5937 },
5938 "subject": {
5939     "anyOf": [
5940         {
5941             "description": "This is the Device identifier.",
5942             "properties": {
5943                 "uuid": {
5944                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5945 4122.",
5946                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5947 fA-F0-9]{12}$",
5948                     "type": "string"
5949                 }
5950             },
5951             "required": [
5952                 "uuid"
5953             ],
5954             "type": "object"
5955         },
5956         {
5957             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
5958 <Authority> refers to the local entity or Device.",
5959             "properties": {
5960                 "authority": {
5961                     "description": "The Authority component of the entity being identified. A
5962 NULL <Authority> refers to the local entity or Device.",
5963                     "type": "string"

```

```

5964         },
5965         "role": {
5966             "description": "The ID of the role being identified.",
5967             "type": "string"
5968         }
5969     },
5970     "required": [
5971         "role"
5972     ],
5973     "type": "object"
5974 },
5975 {
5976     "properties": {
5977         "conntype": {
5978             "description": "This property allows an ACE to be matched based on the
5979 connection or message type.",
5980             "x-detail-desc": [
5981                 "auth-crypt - ACE applies if the Client is authenticated and the data
5982 channel or message is encrypted and integrity protected",
5983                 "anon-clear - ACE applies if the Client is not authenticated and the data
5984 channel or message is not encrypted but may be integrity protected"
5985             ],
5986             "enum": [
5987                 "auth-crypt",
5988                 "anon-clear"
5989             ],
5990             "type": "string"
5991         }
5992     },
5993     "required": [
5994         "conntype"
5995     ],
5996     "type": "object"
5997 }
5998 ]
5999 },
6000 "validity": {
6001     "description": "validity is an array of time-pattern objects.",
6002     "items": {
6003         "description": "The time-pattern contains a period and recurrence expressed in
6004 RFC5545 syntax.",
6005         "properties": {
6006             "period": {
6007                 "description": "String represents a period using the RFC5545 Period.",
6008                 "type": "string"
6009             },
6010             "recurrence": {
6011                 "description": "String array represents a recurrence rule using the RFC5545
6012 Recurrence.",
6013                 "items": {
6014                     "type": "string"
6015                 },
6016                 "type": "array"
6017             }
6018         },
6019         "required": [
6020             "period"
6021         ],
6022         "type": "object"
6023     },
6024     "type": "array"
6025 }
6026 },
6027 "required": [
6028     "aceid",
6029     "resources",
6030     "permission",
6031     "subject"
6032 ],
6033 "type": "object"
6034 },

```

```

6035         "type": "array"
6036     },
6037     "n": {
6038         "$ref":
6039         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6040         schema.json#/definitions/n"
6041     },
6042     "id": {
6043         "$ref":
6044         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6045         schema.json#/definitions/id"
6046     },
6047     "if" : {
6048         "description": "The interface set supported by this Resource.",
6049         "items": {
6050             "enum": [
6051                 "oic.if.baseline"
6052             ],
6053             "type": "string"
6054         },
6055         "minItems": 1,
6056         "maxItems": 1,
6057         "readOnly": true,
6058         "type": "array"
6059     }
6060 },
6061 "type" : "object",
6062 "required": ["aclist2", "rowneruuid"]
6063 },
6064 "Acl2-Update" : {
6065     "properties": {
6066         "rowneruuid" : {
6067             "description": "The value identifies the unique Resource owner\n Format pattern according
6068 to IETF RFC 4122.",
6069             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6070 9]{12}$",
6071             "type": "string"
6072         },
6073         "aclist2" : {
6074             "description": "Access Control Entries in the ACL Resource.",
6075             "items": {
6076                 "properties": {
6077                     "aceid": {
6078 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
6079                     "minimum": 1,
6080                     "type": "integer"
6081                 },
6082             },
6083             "permission": {
6084             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
6085 permissions.",
6086                 "x-detail-desc": [
6087                     "0 - No permissions",
6088                     "1 - Create permission is granted",
6089                     "2 - Read, observe, discover permission is granted",
6090                     "4 - Write, update permission is granted",
6091                     "8 - Delete permission is granted",
6092                     "16 - Notify permission is granted"
6093                 ],
6094                 "maximum": 31,
6095                 "minimum": 0,
6096                 "type": "integer"
6097             },
6098         },
6099         "resources": {
6100         applies.",
6101             "description": "References the application's Resources to which a security policy
6102             applies.",
6103             "items": {
6104                 "description": "Each Resource must have at least one of these properties set.",
6105                 "properties": {
6106                     "href": {
6107                         "description": "When present, the ACE only applies when the href matches\nThis

```

```

6106 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6107     "format": "uri",
6108     "maxLength": 256,
6109     "type": "string"
6110   },
6111   "if": {
6112     "description": "When present, the ACE only applies when the if (interface)
6113 matches\nThe interface set supported by this Resource.",
6114     "items": {
6115       "enum": [
6116         "oic.if.baseline",
6117         "oic.if.ll",
6118         "oic.if.b",
6119         "oic.if.rw",
6120         "oic.if.r",
6121         "oic.if.a",
6122         "oic.if.s"
6123       ],
6124       "type": "string"
6125     },
6126     "minItems": 1,
6127     "type": "array"
6128   },
6129   "rt": {
6130     "description": "When present, the ACE only applies when the rt (Resource type)
6131 matches\nResource Type of the Resource.",
6132     "items": {
6133       "maxLength": 64,
6134       "type": "string"
6135     },
6136     "minItems": 1,
6137     "type": "array"
6138   },
6139   "wc": {
6140     "description": "A wildcard matching policy.",
6141     "x-detail-desc": [
6142       "+ - Matches all discoverable Resources",
6143       "- - Matches all non-discoverable Resources",
6144       "* - Matches all Resources"
6145     ],
6146     "enum": [
6147       "+",
6148       "-",
6149       "*"
6150     ],
6151     "type": "string"
6152   }
6153 },
6154 "type": "object"
6155 },
6156 "type": "array"
6157 },
6158 "subject": {
6159   "anyOf": [
6160     {
6161       "description": "This is the Device identifier.",
6162       "properties": {
6163         "uuid": {
6164           "description": "A UUID Device ID\n Format pattern according to IETF RFC
6165 4122.",
6166           "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
6167 fA-F0-9]{12}$",
6168           "type": "string"
6169         }
6170       },
6171       "required": [
6172         "uuid"
6173       ],
6174       "type": "object"
6175     },
6176     {

```

```

6177         "description": "Security role specified as an <Authority> & <Rolename>. A NULL
6178 <Authority> refers to the local entity or Device.",
6179         "properties": {
6180             "authority": {
6181                 "description": "The Authority component of the entity being identified. A
6182 NULL <Authority> refers to the local entity or Device.",
6183                 "type": "string"
6184             },
6185             "role": {
6186                 "description": "The ID of the role being identified.",
6187                 "type": "string"
6188             }
6189         },
6190         "required": [
6191             "role"
6192         ],
6193         "type": "object"
6194     },
6195     {
6196         "properties": {
6197             "conntype": {
6198                 "description": "This property allows an ACE to be matched based on the
6199 connection or message type.",
6200                 "x-detail-desc": [
6201                     "auth-crypt - ACE applies if the Client is authenticated and the data
6202 channel or message is encrypted and integrity protected",
6203                     "anon-clear - ACE applies if the Client is not authenticated and the data
6204 channel or message is not encrypted but may be integrity protected"
6205                 ],
6206                 "enum": [
6207                     "auth-crypt",
6208                     "anon-clear"
6209                 ],
6210                 "type": "string"
6211             }
6212         },
6213         "required": [
6214             "conntype"
6215         ],
6216         "type": "object"
6217     }
6218 ]
6219 },
6220 "validity": {
6221     "description": "validity is an array of time-pattern objects.",
6222     "items": {
6223         "description": "The time-pattern contains a period and recurrence expressed in
6224 RFC5545 syntax.",
6225         "properties": {
6226             "period": {
6227                 "description": "String represents a period using the RFC5545 Period.",
6228                 "type": "string"
6229             },
6230             "recurrence": {
6231                 "description": "String array represents a recurrence rule using the RFC5545
6232 Recurrence.",
6233                 "items": {
6234                     "type": "string"
6235                 },
6236                 "type": "array"
6237             }
6238         },
6239         "required": [
6240             "period"
6241         ],
6242         "type": "object"
6243     },
6244     "type": "array"
6245 }
6246 },
6247 "required": [

```

```

6248         "resources",
6249         "permission",
6250         "subject"
6251     ],
6252     "type": "object"
6253 },
6254 "type": "array"
6255 },
6256 },
6257 "type" : "object"
6258 }
6259 }
6260 }
6261 }

```

#### 6262 C.4.5 Property definition

6263 Table C.6 defines the Properties that are part of the "oic.r.acl2" Resource Type.

6264 **Table C.6 – The Property definitions of the Resource with type "rt" = "oic.r.acl2".**

Property name	Value type	Mandatory	Access mode	Description
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.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL Resource.
rowneruuid	string	Yes	Read Write	The value identifies the unique Resource owner Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

#### 6265 C.4.6 CRUDN behaviour

6266 Table C.7 defines the CRUDN operations that are supported on the "oic.r.acl2" Resource Type.

6267 **Table C.7 – The CRUDN operations of the Resource with type "rt" = "oic.r.acl2".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 6268 C.5 Managed Access Control

### 6269 C.5.1 Introduction

6270 This Resource specifies the host Resources with access permission that is managed by an AMS.

### 6271 C.5.2 Well-known URI

6272 /oic/sec/amacl

### 6273 C.5.3 Resource type

6274 The Resource Type is defined as: "oic.r.amacl".

### 6275 C.5.4 OpenAPI 2.0 definition

```
6276 {
6277   "swagger": "2.0",
6278   "info": {
6279     "title": "Managed Access Control",
6280     "version": "20190111",
6281     "license": {
6282       "name": "OCF Data Model License",
6283       "url":
6284         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6285         CENSE.md",
6286       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6287         reserved."
6288     },
6289     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6290   },
6291   "schemes": ["http"],
6292   "consumes": ["application/json"],
6293   "produces": ["application/json"],
6294   "paths": {
6295     "/oic/sec/amacl" : {
6296       "get": {
6297         "description": "This Resource specifies the host Resources with access permission that is
6298         managed by an AMS.\n",
6299         "parameters": [
6300           {"$ref": "#/parameters/interface"}
6301         ],
6302         "responses": {
6303           "200": {
6304             "description": "",
6305             "x-example":
6306               {
6307                 "rt" : ["oic.r.amacl"],
6308                 "resources": [
6309                   {
6310                     "href": "/temp",
6311                     "rt": ["oic.r.temperature"],
6312                     "if": ["oic.if.baseline", "oic.if.a"]
6313                   },
6314                   {
6315                     "href": "/temp",
6316                     "rt": ["oic.r.temperature"],
6317                     "if": ["oic.if.baseline", "oic.if.s"]
6318                   }
6319                 ]
6320             },
6321           "schema": { "$ref": "#/definitions/Amacl" }
6322         }
6323       }
6324     },
6325     "post": {
6326       "description": "Sets the new amacl data.\n",
6327       "parameters": [
6328         {"$ref": "#/parameters/interface"},
6329         {
6330           "name": "body",
6331           "in": "body",

```

```

6332         "required": true,
6333         "schema": { "$ref": "#/definitions/Amacl" },
6334         "x-example":
6335             {
6336                 "resources": [
6337                     {
6338                         "href": "/temp",
6339                         "rt": ["oic.r.temperature"],
6340                         "if": ["oic.if.baseline", "oic.if.a"]
6341                     },
6342                     {
6343                         "href": "/temp",
6344                         "rt": ["oic.r.temperature"],
6345                         "if": ["oic.if.baseline", "oic.if.s"]
6346                     }
6347                 ]
6348             }
6349     ],
6350     "responses": {
6351         "400": {
6352             "description": "The request is invalid."
6353         },
6354         "201": {
6355             "description": "The AMACL entry is created."
6356         },
6357         "204": {
6358             "description": "The AMACL entry is updated."
6359         }
6360     }
6361 },
6362 "put": {
6363     "description": "Creates the new acl data.\n",
6364     "parameters": [
6365         { "$ref": "#/parameters/interface" },
6366         {
6367             "name": "body",
6368             "in": "body",
6369             "required": true,
6370             "schema": { "$ref": "#/definitions/Amacl" },
6371             "x-example":
6372                 {
6373                     "resources": [
6374                         {
6375                             "href": "/temp",
6376                             "rt": ["oic.r.temperature"],
6377                             "if": ["oic.if.baseline", "oic.if.a"]
6378                         },
6379                         {
6380                             "href": "/temp",
6381                             "rt": ["oic.r.temperature"],
6382                             "if": ["oic.if.baseline", "oic.if.s"]
6383                         }
6384                     ]
6385                 }
6386             }
6387     ],
6388     "responses": {
6389         "400": {
6390             "description": "The request is invalid."
6391         },
6392         "201": {
6393             "description": "The AMACL entry is created."
6394         }
6395     }
6396 },
6397 "delete": {
6398     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
6399     entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
6400     matched entry is deleted.\n",
6401     "parameters": [
6402

```



```

6403         {"$ref": "#/parameters/interface"},
6404         {
6405             "in": "query",
6406             "description": "Delete the ACE identified by the string matching the subject value.\n",
6407             "type": "string",
6408             "name": "subject"
6409         }
6410     ],
6411     "responses": {
6412         "200": {
6413             "description": "The ACE instance or the the entire AMACL Resource has been
6414 successfully deleted."
6415         },
6416         "400": {
6417             "description": "The request is invalid."
6418         }
6419     }
6420 }
6421 }
6422 },
6423 "parameters": {
6424     "interface" : {
6425         "in" : "query",
6426         "name" : "if",
6427         "type" : "string",
6428         "enum" : ["oic.if.baseline"]
6429     }
6430 },
6431 "definitions": {
6432     "Amacl" : {
6433         "properties": {
6434             "rt" : {
6435                 "description": "Resource Type of the Resource.",
6436                 "items": {
6437                     "maxLength": 64,
6438                     "type": "string",
6439                     "enum": ["oic.r.amacl"]
6440                 },
6441                 "minItems": 1,
6442                 "maxItems": 1,
6443                 "readOnly": true,
6444                 "type": "array"
6445             },
6446             "n": {
6447                 "$ref":
6448 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6449 schema.json#/definitions/n"
6450             },
6451             "id": {
6452                 "$ref":
6453 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6454 schema.json#/definitions/id"
6455             },
6456             "resources" : {
6457                 "description": "Multiple links to this host's Resources.",
6458                 "items": {
6459                     "description": "Each Resource must have at least one of these properties set.",
6460                     "properties": {
6461                         "href": {
6462                             "description": "When present, the ACE only applies when the href matches\nThis is
6463 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6464                             "format": "uri",
6465                             "maxLength": 256,
6466                             "type": "string"
6467                         },
6468                         "if": {
6469                             "description": "When present, the ACE only applies when the if (interface)
6470 matches\nThe interface set supported by this Resource.",
6471                             "items": {
6472                                 "enum": [
6473                                     "oic.if.baseline",

```

```

6474         "oic.if.ll",
6475         "oic.if.b",
6476         "oic.if.rw",
6477         "oic.if.r",
6478         "oic.if.a",
6479         "oic.if.s"
6480     ],
6481     "type": "string"
6482 },
6483     "minItems": 1,
6484     "type": "array"
6485 },
6486     "rt": {
6487         "description": "When present, the ACE only applies when the rt (Resource type)
6488 matches\nResource Type of the Resource.",
6489         "items": {
6490             "maxLength": 64,
6491             "type": "string"
6492         },
6493         "minItems": 1,
6494         "type": "array"
6495     },
6496     "wc": {
6497         "description": "A wildcard matching policy.",
6498         "pattern": "^[~+]*$",
6499         "type": "string"
6500     }
6501 },
6502     "type": "object"
6503 },
6504     "type": "array"
6505 },
6506     "if" : {
6507         "description": "The interface set supported by this Resource.",
6508         "items": {
6509             "enum": [
6510                 "oic.if.baseline"
6511             ],
6512             "type": "string"
6513         },
6514         "minItems": 1,
6515         "maxItems": 1,
6516         "readOnly": true,
6517         "type": "array"
6518     }
6519 },
6520     "type" : "object",
6521     "required": ["resources"]
6522 }
6523 }
6524 }
6525

```

### 6526 C.5.5 Property definition

6527 Table C.8 defines the Properties that are part of the "oic.r.amacl" Resource Type.

6528 **Table C.8– The Property definitions of the Resource with type "rt" = "oic.r.amacl".**

Property name	Value type	Mandatory	Access mode	Description
resources	array: see schema	Yes	Read Write	Multiple links to this host's Resources.
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.

rt	array: see schema	No	Read Only	Resource Type of the Resource.
id	multiple types: see schema	No	Read Write	

6529 **C.5.6 CRUDN behaviour**

6530 Table C.9 defines the CRUDN operations that are supported on the "oic.r.amacl" Resource Type.

6531 **Table C.9 – The CRUDN operations of the Resource with type "rt" = "oic.r.amacl".**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

6532 **C.6 Credential**

6533 **C.6.1 Introduction**

6534 This Resource specifies credentials a Device may use to establish secure communication.

6535 Retrieves the credential data.

6536 When used without query parameters, all the credential entries are returned.

6537 When used with a query parameter, only the credentials matching the specified  
6538 parameter are returned.

6539

6540 Note that write-only credential data will not be returned.

6541

6542 **C.6.2 Well-known URI**

6543 /oic/sec/cred

6544 **C.6.3 Resource type**

6545 The Resource Type is defined as: "oic.r.cred".

6546 **C.6.4 OpenAPI 2.0 definition**

```
6547 {
6548   "swagger": "2.0",
6549   "info": {
6550     "title": "Credential",
6551     "version": "v1.0-20181031",
6552     "license": {
6553       "name": "OCF Data Model License",
6554       "url":
6555       "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6556       CENSE.md",
6557       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6558       reserved."
6559     },
6560     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6561   },
6562   "schemes": ["http"],
6563   "consumes": ["application/json"],
6564   "produces": ["application/json"],
6565   "paths": {
6566     "/oic/sec/cred" : {
6567       "get": {
6568         "description": "This Resource specifies credentials a Device may use to establish secure
6569         communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
6570         credential entries are returned.\nWhen used with a query parameter, only the credentials matching
6571         the specified\nparameter are returned.\n\nNote that write-only credential data will not be
6572         returned.\n",
6573         "parameters": [
6574           {"$ref": "#/parameters/interface"}
6575           ,{"$ref": "#/parameters/cred-filtered-credid"}
6576           ,{"$ref": "#/parameters/cred-filtered-subjectuuid"}
6577         ],

```

```

6578     "responses": {
6579         "200": {
6580             "description": "",
6581             "x-example":
6582                 {
6583                     "rt": ["oic.r.cred"],
6584                     "creds": [
6585                         {
6586                             "credid": 55,
6587                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6588                             "roleid": {
6589                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6590                                 "role": "SOME_STRING"
6591                             },
6592                             "credtype": 32,
6593                             "publicdata": {
6594                                 "encoding": "oic.sec.encoding.base64",
6595                                 "data": "BASE-64-ENCODED-VALUE"
6596                             },
6597                             "privatedata": {
6598                                 "encoding": "oic.sec.encoding.base64",
6599                                 "data": "BASE-64-ENCODED-VALUE",
6600                                 "handle": 4
6601                             },
6602                             "optionaldata": {
6603                                 "revstat": false,
6604                                 "encoding": "oic.sec.encoding.base64",
6605                                 "data": "BASE-64-ENCODED-VALUE"
6606                             },
6607                             "period": "20160101T180000Z/20170102T070000Z",
6608                             "crms": [ "oic.sec.crm.pk10" ]
6609                         },
6610                         {
6611                             "credid": 56,
6612                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6613                             "roleid": {
6614                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6615                                 "role": "SOME_STRING"
6616                             },
6617                             "credtype": 1,
6618                             "publicdata": {
6619                                 "encoding": "oic.sec.encoding.base64",
6620                                 "data": "BASE-64-ENCODED-VALUE"
6621                             },
6622                             "privatedata": {
6623                                 "encoding": "oic.sec.encoding.base64",
6624                                 "data": "BASE-64-ENCODED-VALUE",
6625                                 "handle": 4
6626                             },
6627                             "optionaldata": {
6628                                 "revstat": false,
6629                                 "encoding": "oic.sec.encoding.base64",
6630                                 "data": "BASE-64-ENCODED-VALUE"
6631                             },
6632                             "period": "20160101T180000Z/20170102T070000Z",
6633                             "crms": [ "oic.sec.crm.pk10" ]
6634                         }
6635                     ],
6636                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6637                 }
6638             ,
6639             "schema": { "$ref": "#/definitions/Cred" }
6640         },
6641         "400": {
6642             "description": "The request is invalid."
6643         }
6644     },
6645     "post": {
6646         "description": "Updates the credential Resource with the provided
6647         credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential

```

```

6649 Resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
6650 Resource completely replace the creds in the credential\nResource.\n\nCredentials provided in the
6651 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
6652 Resource.\n",
6653     "parameters": [
6654         { "$ref": "#/parameters/interface" },
6655     ],
6656     "name": "body",
6657     "in": "body",
6658     "required": true,
6659     "schema": { "$ref": "#/definitions/Cred-Update" },
6660     "x-example":
6661     {
6662         "creds": [
6663             {
6664                 "credid": 55,
6665                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6666                 "roleid": {
6667                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6668                     "role": "SOME_STRING"
6669                 },
6670                 "credtype": 32,
6671                 "publicdata": {
6672                     "encoding": "oic.sec.encoding.base64",
6673                     "data": "BASE-64-ENCODED-VALUE"
6674                 },
6675                 "privatedata": {
6676                     "encoding": "oic.sec.encoding.base64",
6677                     "data": "BASE-64-ENCODED-VALUE",
6678                     "handle": 4
6679                 },
6680                 "optionaldata": {
6681                     "revstat": false,
6682                     "encoding": "oic.sec.encoding.base64",
6683                     "data": "BASE-64-ENCODED-VALUE"
6684                 },
6685                 "period": "20160101T180000Z/20170102T070000Z",
6686                 "crms": [ "oic.sec.crm.pk10" ]
6687             },
6688             {
6689                 "credid": 56,
6690                 "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6691                 "roleid": {
6692                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6693                     "role": "SOME_STRING"
6694                 },
6695                 "credtype": 1,
6696                 "publicdata": {
6697                     "encoding": "oic.sec.encoding.base64",
6698                     "data": "BASE-64-ENCODED-VALUE"
6699                 },
6700                 "privatedata": {
6701                     "encoding": "oic.sec.encoding.base64",
6702                     "data": "BASE-64-ENCODED-VALUE",
6703                     "handle": 4
6704                 },
6705                 "optionaldata": {
6706                     "revstat": false,
6707                     "encoding": "oic.sec.encoding.base64",
6708                     "data": "BASE-64-ENCODED-VALUE"
6709                 },
6710                 "period": "20160101T180000Z/20170102T070000Z",
6711                 "crms": [ "oic.sec.crm.pk10" ]
6712             }
6713         ],
6714         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6715     }
6716 ],
6717 "responses": {
6718     "400": {
6719

```

```

6720         "description" : "The request is invalid."
6721     },
6722     "201": {
6723         "description" : "The credential entry is created."
6724     },
6725     "204": {
6726         "description" : "The credential entry is updated."
6727     }
6728 },
6729 },
6730     "delete": {
6731         "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
6732 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
6733 matching\nthe query parameter are deleted.\n",
6734         "parameters": [
6735             {"$ref": "#/parameters/interface"},
6736             {"$ref": "#/parameters/cred-filtered-credid"},
6737             {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6738         ],
6739         "responses": {
6740             "400": {
6741                 "description" : "The request is invalid."
6742             },
6743             "204": {
6744                 "description" : "The specific credential(s) or the the entire credential Resource has
6745 been successfully deleted."
6746             }
6747         }
6748     }
6749 },
6750 },
6751 "parameters": {
6752     "interface" : {
6753         "in" : "query",
6754         "name" : "if",
6755         "type" : "string",
6756         "enum" : ["oic.if.baseline"]
6757     },
6758     "cred-filtered-credid" : {
6759         "in" : "query",
6760         "name" : "credid",
6761         "required" : false,
6762         "type" : "integer",
6763         "description" : "Only applies to the credential with the specified credid.",
6764         "x-example" : 2112
6765     },
6766     "cred-filtered-subjectuuid" : {
6767         "in" : "query",
6768         "name" : "subjectuuid",
6769         "required" : false,
6770         "type" : "string",
6771         "description" : "Only applies to credentials with the specified subject UUID.",
6772         "x-example" : "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6773     }
6774 },
6775 "definitions": {
6776     "Cred" : {
6777         "properties": {
6778             "rowneruuid" : {
6779                 "description": "Format pattern according to IETF RFC 4122.",
6780                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
6781                 "type": "string"
6782             },
6783             "rt" : {
6784                 "description": "Resource Type of the Resource.",
6785                 "items": {
6786                     "maxLength": 64,
6787                     "type": "string",
6788                     "enum": ["oic.r.cred"]
6789                 },
6790                 "minItems": 1,

```

```

6791         "readOnly": true,
6792         "type": "array",
6793         "uniqueItems": true
6794     },
6795     "n": {
6796         "$ref":
6797 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6798 schema.json#/definitions/n"
6799     },
6800     "id": {
6801         "$ref":
6802 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6803 schema.json#/definitions/id"
6804     },
6805     "creds" : {
6806         "description": "List of credentials available at this Resource.",
6807         "items": {
6808             "properties": {
6809                 "credid": {
6810                     "description": "Local reference to a credential Resource.",
6811                     "type": "integer"
6812                 },
6813                 "credtype": {
6814                     "description": "Representation of this credential's type\nCredential Types - Cred
6815 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6816 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
6817 password32 - Asymmetric encryption key.",
6818                     "maximum": 63,
6819                     "minimum": 0,
6820                     "type": "integer"
6821                 },
6822                 "credusage": {
6823                     "description": "A string that provides hints about how/where the cred is used\nThe
6824 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6825 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6826 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
6827                     "enum": [
6828                         "oic.sec.cred.trustca",
6829                         "oic.sec.cred.cert",
6830                         "oic.sec.cred.rolecert",
6831                         "oic.sec.cred.mfgtrustca",
6832                         "oic.sec.cred.mfgcert"
6833                     ],
6834                     "type": "string"
6835                 },
6836                 "crms": {
6837                     "description": "The refresh methods that may be used to update this credential.",
6838                     "items": {
6839                         "description": "Each enum represents a method by which the credentials are
6840 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6841 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6842 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6843 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
6844                     "enum": [
6845                         "oic.sec.crm.pro",
6846                         "oic.sec.crm.psk",
6847                         "oic.sec.crm.rdp",
6848                         "oic.sec.crm.skdc",
6849                         "oic.sec.crm.pk10"
6850                     ],
6851                     "type": "string"
6852                 },
6853                 "type": "array",
6854                 "uniqueItems" : true
6855             },
6856             "optionaldata": {
6857                 "description": "Credential revocation status information\nOptional credential
6858 contents describes revocation status for this credential.",
6859                 "properties": {
6860                     "data": {
6861                         "description": "The encoded structure.",

```

```

6862         "type": "string"
6863     },
6864     "encoding": {
6865         "description": "A string specifying the encoding format of the data contained in
6866 the optdata.",
6867         "x-detail-desc": [
6868             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
6869             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
6870             "oic.sec.encoding.base64 - Base64 encoded object.",
6871             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
6872             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
6873             "oic.sec.encoding.raw - Raw hex encoded data."
6874         ],
6875         "enum": [
6876             "oic.sec.encoding.jwt",
6877             "oic.sec.encoding.cwt",
6878             "oic.sec.encoding.base64",
6879             "oic.sec.encoding.pem",
6880             "oic.sec.encoding.der",
6881             "oic.sec.encoding.raw"
6882         ],
6883         "type": "string"
6884     },
6885     "revstat": {
6886         "description": "Revocation status flag - true = revoked.",
6887         "type": "boolean"
6888     }
6889 },
6890 "required": [
6891     "revstat"
6892 ],
6893 "type": "object"
6894 },
6895 "period": {
6896     "description": "String with RFC5545 Period.",
6897     "type": "string"
6898 },
6899 "privatedata": {
6900     "description": "Private credential information\nCredential Resource non-public
6901 contents.",
6902     "properties": {
6903         "data": {
6904             "description": "The encoded value.",
6905             "maxLength": 3072,
6906             "type": "string"
6907         },
6908         "encoding": {
6909             "description": "A string specifying the encoding format of the data contained in
6910 the privdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6911 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6912 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6913 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data.",
6914             "enum": [
6915                 "oic.sec.encoding.jwt",
6916                 "oic.sec.encoding.cwt",
6917                 "oic.sec.encoding.base64",
6918                 "oic.sec.encoding.uri",
6919                 "oic.sec.encoding.handle",
6920                 "oic.sec.encoding.raw"
6921             ],
6922             "type": "string"
6923         },
6924         "handle": {
6925             "description": "Handle to a key storage Resource.",
6926             "type": "integer"
6927         }
6928     },
6929     "required": [
6930         "encoding"
6931     ],
6932     "type": "object"

```



```

6933     },
6934     "publicdata": {
6935         "description": "Public credential information.",
6936         "properties": {
6937             "data": {
6938                 "description": "The encoded value.",
6939                 "maxLength": 3072,
6940                 "type": "string"
6941             },
6942             "encoding": {
6943                 "description": "A string specifying the encoding format of the data contained in
6944 the pubdata\noic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding\noic.sec.encoding.cwt -
6945 RFC CBOR web token (CWT) encoding\noic.sec.encoding.base64 - Base64 encoded
6946 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.pem - Encoding for PEM encoded
6947 certificate or chain\noic.sec.encoding.der - Encoding for DER encoded
6948 certificate\noic.sec.encoding.raw - Raw hex encoded data.",
6949                 "enum": [
6950                     "oic.sec.encoding.jwt",
6951                     "oic.sec.encoding.cwt",
6952                     "oic.sec.encoding.base64",
6953                     "oic.sec.encoding.uri",
6954                     "oic.sec.encoding.pem",
6955                     "oic.sec.encoding.der",
6956                     "oic.sec.encoding.raw"
6957                 ],
6958                 "type": "string"
6959             }
6960         },
6961         "type": "object"
6962     },
6963     "roleid": {
6964         "description": "The role this credential possesses\nSecurity role specified as an
6965 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
6966         "properties": {
6967             "authority": {
6968                 "description": "The Authority component of the entity being identified. A NULL
6969 <Authority> refers to the local entity or Device.",
6970                 "type": "string"
6971             },
6972             "role": {
6973                 "description": "The ID of the role being identified.",
6974                 "type": "string"
6975             }
6976         },
6977         "required": [
6978             "role"
6979         ],
6980         "type": "object"
6981     },
6982     "subjectuuid": {
6983         "anyOf": [
6984             {
6985                 "description": "The id of the Device, which the cred entry applies to or \"*\n"
6986 for wildcard identity.",
6987                 "pattern": "^\\*$",
6988                 "type": "string"
6989             },
6990             {
6991                 "description": "Format pattern according to IETF RFC 4122.",
6992                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
6993 F0-9]{12}$",
6994                 "type": "string"
6995             }
6996         ]
6997     }
6998 },
6999 "type": "object"
7000 },
7001 "type": "array"
7002 },
7003 "if" : {

```

```

7004         "description": "The interface set supported by this Resource.",
7005         "items": {
7006             "enum": [
7007                 "oic.if.baseline"
7008             ],
7009             "type": "string"
7010         },
7011         "minItems": 1,
7012         "readOnly": true,
7013         "type": "array"
7014     }
7015 },
7016 "type" : "object",
7017 "required": ["creds", "rowneruuid"]
7018 },
7019 "Cred-Update" : {
7020     "properties": {
7021         "rowneruuid" : {
7022             "description": "Format pattern according to IETF RFC 4122.",
7023             "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}$",
7024             "type": "string"
7025         },
7026         "creds" : {
7027             "description": "List of credentials available at this Resource.",
7028             "items": {
7029                 "properties": {
7030                     "credid": {
7031                         "description": "Local reference to a credential Resource.",
7032                         "type": "integer"
7033                     },
7034                     "credtype": {
7035                         "description": "Representation of this credential's type\nCredential Types - Cred
7036 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7037 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7038 password32 - Asymmetric encryption key.",
7039                         "maximum": 63,
7040                         "minimum": 0,
7041                         "type": "integer"
7042                     },
7043                     "credusage": {
7044                         "description": "A string that provides hints about how/where the cred is used\nThe
7045 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7046 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7047 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
7048                         "enum": [
7049                             "oic.sec.cred.trustca",
7050                             "oic.sec.cred.cert",
7051                             "oic.sec.cred.rolecert",
7052                             "oic.sec.cred.mfgtrustca",
7053                             "oic.sec.cred.mfgcert"
7054                         ],
7055                         "type": "string"
7056                     },
7057                     "crms": {
7058                         "description": "The refresh methods that may be used to update this credential.",
7059                         "items": {
7060                             "description": "Each enum represents a method by which the credentials are
7061 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7062 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7063 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
7064 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
7065                             "enum": [
7066                                 "oic.sec.crm.pro",
7067                                 "oic.sec.crm.psk",
7068                                 "oic.sec.crm.rdp",
7069                                 "oic.sec.crm.skdc",
7070                                 "oic.sec.crm.pk10"
7071                             ],
7072                             "type": "string"
7073                         },
7074                     },
7075                     "type": "array"

```

```

7075     },
7076     "optionaldata": {
7077         "description": "Credential revocation status information\nOptional credential
7078 contents describes revocation status for this credential.",
7079         "properties": {
7080             "data": {
7081                 "description": "The encoded structure.",
7082                 "type": "string"
7083             },
7084             "encoding": {
7085                 "description": "A string specifying the encoding format of the data contained in
7086 the optdata.",
7087                 "x-detail-desc": [
7088                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7089                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7090                     "oic.sec.encoding.base64 - Base64 encoded object.",
7091                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7092                     "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7093                     "oic.sec.encoding.raw - Raw hex encoded data."
7094                 ],
7095                 "enum": [
7096                     "oic.sec.encoding.jwt",
7097                     "oic.sec.encoding.cwt",
7098                     "oic.sec.encoding.base64",
7099                     "oic.sec.encoding.pem",
7100                     "oic.sec.encoding.der",
7101                     "oic.sec.encoding.raw"
7102                 ],
7103                 "type": "string"
7104             },
7105             "revstat": {
7106                 "description": "Revocation status flag - true = revoked.",
7107                 "type": "boolean"
7108             }
7109         },
7110         "required": [
7111             "revstat"
7112         ],
7113         "type" : "object"
7114     },
7115     "period": {
7116         "description": "String with RFC5545 Period.",
7117         "type": "string"
7118     },
7119     "privatedata": {
7120         "description": "Private credential information\nCredential Resource non-public
7121 contents.",
7122         "properties": {
7123             "data": {
7124                 "description": "The encoded value.",
7125                 "maxLength": 3072,
7126                 "type": "string"
7127             },
7128             "encoding": {
7129                 "description": "A string specifying the encoding format of the data contained in
7130 the privdata.",
7131                 "x-detail-desc": [
7132                     "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7133                     "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7134                     "oic.sec.encoding.base64 - Base64 encoded object.",
7135                     "oic.sec.encoding.uri - URI reference.",
7136                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
7137 referenced using a handle.",
7138                     "oic.sec.encoding.raw - Raw hex encoded data."
7139                 ],
7140                 "enum": [
7141                     "oic.sec.encoding.jwt",
7142                     "oic.sec.encoding.cwt",
7143                     "oic.sec.encoding.base64",
7144                     "oic.sec.encoding.uri",
7145                     "oic.sec.encoding.handle",

```

```

7146         "oic.sec.encoding.raw"
7147     ],
7148     "type": "string"
7149 },
7150 "handle": {
7151     "description": "Handle to a key storage Resource.",
7152     "type": "integer"
7153 },
7154 },
7155 "required": [
7156     "encoding"
7157 ],
7158 "type": "object"
7159 },
7160 "publicdata": {
7161     "properties": {
7162         "data": {
7163             "description": "The encoded value.",
7164             "maxLength": 3072,
7165             "type": "string"
7166         },
7167         "encoding": {
7168             "description": "Public credential information\nA string specifying the encoding
7169 format of the data contained in the pubdata.",
7170             "x-detail-desc": [
7171                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
7172                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
7173                 "oic.sec.encoding.base64 - Base64 encoded object.",
7174                 "oic.sec.encoding.uri - URI reference.",
7175                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
7176                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
7177                 "oic.sec.encoding.raw - Raw hex encoded data."
7178             ],
7179             "enum": [
7180                 "oic.sec.encoding.jwt",
7181                 "oic.sec.encoding.cwt",
7182                 "oic.sec.encoding.base64",
7183                 "oic.sec.encoding.uri",
7184                 "oic.sec.encoding.pem",
7185                 "oic.sec.encoding.der",
7186                 "oic.sec.encoding.raw"
7187             ],
7188             "type": "string"
7189         }
7190     },
7191     "type": "object"
7192 },
7193 "roleid": {
7194     "description": "The role this credential possesses\nSecurity role specified as an
7195 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
7196     "properties": {
7197         "authority": {
7198             "description": "The Authority component of the entity being identified. A NULL
7199 <Authority> refers to the local entity or Device.",
7200             "type": "string"
7201         },
7202         "role": {
7203             "description": "The ID of the role being identified.",
7204             "type": "string"
7205         }
7206     },
7207     "required": [
7208         "role"
7209 ],
7210     "type": "object"
7211 },
7212 "subjectuuid": {
7213     "anyOf": [
7214         {
7215             "description": "The id of the Device, which the cred entry applies to or \"*\n
7216 for wildcard identity.",

```

```

7217         "pattern": "^\\*$",
7218         "type": "string"
7219     },
7220     {
7221         "description": "Format pattern according to IETF RFC 4122.",
7222         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7223 F0-9]{12}$",
7224         "type": "string"
7225     }
7226 ]
7227 },
7228 },
7229 "type": "object"
7230 },
7231 "type": "array"
7232 },
7233 "if" :
7234 {
7235     "description": "The interface set supported by this Resource.",
7236     "items": {
7237         "enum": [
7238             "oic.if.baseline"
7239         ],
7240         "type": "string"
7241     },
7242     "minItems": 1,
7243     "readOnly": true,
7244     "type": "array"
7245 }
7246 },
7247 "type" : "object"
7248 }
7249 }
7250 }
7251

```

### 7252 C.6.5 Property definition

7253 Table C.10 defines the Properties that are part of the "oic.r.cred" Resource Type.

7254 **Table C.10 – 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 interface set supported by this Resource.
creds	array: see schema	No	Read Write	List of credentials available at this Resource.
id	multiple types: see schema	No	Read Write	
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
n	multiple types: see schema	No	Read Write	

creds	array: schema	see	Yes	Read Write	List of credentials available at this Resource.
-------	------------------	-----	-----	------------	--

7255 **C.6.6 CRUDN behaviour**

7256 Table C.11 defines the CRUDN operations that are supported on the "oic.r.cred" Resource Type.

7257 **Table C.11 – The CRUDN operations of the Resource with type "rt" = "oic.r.cred".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

7258 **C.7 Certificate Revocation**

7259 **C.7.1 Introduction**

7260 This Resource specifies certificate revocation lists as X.509 objects.

7261 **C.7.2 Well-known URI**

7262 /oic/sec/crl

7263 **C.7.3 Resource type**

7264 The Resource Type is defined as: "oic.r.crl".

7265 **C.7.4 OpenAPI 2.0 definition**

```

7266 {
7267   "swagger": "2.0",
7268   "info": {
7269     "title": "Certificate Revocation",
7270     "version": "v1.0-20150819",
7271     "license": {
7272       "name": "OCF Data Model License",
7273       "url":
7274         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7275         CENSE.md",
7276       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7277         reserved."
7278     },
7279     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7280   },
7281   "schemes": ["http"],
7282   "consumes": ["application/json"],
7283   "produces": ["application/json"],
7284   "paths": {
7285     "/oic/sec/crl" : {
7286       "get": {
7287         "description": "This Resource specifies certificate revocation lists as X.509 objects.\n",
7288         "parameters": [
7289           { "$ref": "#/parameters/interface" }
7290         ],
7291         "responses": {
7292           "200": {
7293             "description": "",
7294             "x-example":
7295               {
7296                 "rt": ["oic.r.crl"],
7297                 "crlid": 1,
7298                 "thisupdate": "2016-04-12T23:20:50.52Z",
7299                 "crldata": "Base64ENCODEDCRL"
7300               },
7301             "schema": { "$ref": "#/definitions/Crl" }
7302           }
7303         }
7304       },
7305       "post": {

```

```

7306     "description": "Updates the CRL data.\n",
7307     "parameters": [
7308       { "$ref": "#/parameters/interface" },
7309       {
7310         "name": "body",
7311         "in": "body",
7312         "required": true,
7313         "schema": { "$ref": "#/definitions/Crl-Update" },
7314         "x-example":
7315           {
7316             "crlid": 1,
7317             "thisupdate": "2016-04-12T23:20:50.52Z",
7318             "crldata": "Base64ENCODEDCRL"
7319           }
7320       }
7321     ],
7322     "responses": {
7323       "400": {
7324         "description": "The request is invalid."
7325       },
7326       "204": {
7327         "description": "The CRL entry is updated."
7328       }
7329     }
7330   }
7331 },
7332 },
7333 "parameters": {
7334   "interface" : {
7335     "in" : "query",
7336     "name" : "if",
7337     "type" : "string",
7338     "enum" : ["oic.if.baseline"]
7339   }
7340 },
7341 "definitions": {
7342   "Crl" : {
7343     "properties": {
7344       "rt" : {
7345         "description": "Resource Type of the Resource.",
7346         "items": {
7347           "maxLength": 64,
7348           "type": "string",
7349           "enum": ["oic.r.crl"]
7350         },
7351         "minItems": 1,
7352         "readOnly": true,
7353         "type": "array"
7354       },
7355       "n": {
7356         "$ref":
7357 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7358 schema.json#/definitions/n"
7359       },
7360       "id": {
7361         "$ref":
7362 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7363 schema.json#/definitions/id"
7364       },
7365       "crldata" : {
7366         "description": "Base64 BER encoded CRL data.",
7367         "type": "string"
7368       },
7369       "crlid" : {
7370         "description": "Local reference to a CRL Resource.",
7371         "type": "integer"
7372       },
7373       "thisupdate" : {
7374         "description": "UTC time of last CRL update.",
7375         "type": "string"
7376     },

```

```

7377     "if" : {
7378         "description": "The interface set supported by this Resource.",
7379         "items": {
7380             "enum": [
7381                 "oic.if.baseline"
7382             ],
7383             "type": "string"
7384         },
7385         "minItems": 1,
7386         "readOnly": true,
7387         "type": "array"
7388     },
7389 },
7390 "type": "object",
7391 "required": ["crlid", "thisupdate", "crldata"]
7392 }
7393 ,
7394 "Crl-Update": {
7395     "properties": {
7396         "crldata": {
7397             "description": "Base64 BER encoded CRL data.",
7398             "type": "string"
7399         },
7400         "crlid": {
7401             "description": "Local reference to a CRL Resource.",
7402             "type": "integer"
7403         },
7404         "thisupdate": {
7405             "description": "UTC time of last CRL update.",
7406             "type": "string"
7407         }
7408     },
7409     "type" : "object"
7410 }
7411 }
7412 }
7413

```

### 7414 C.7.5 Property definition

7415 Table C.12 defines the Properties that are part of the "oic.r.crl" Resource Type.

7416 **Table C.12 – The Property definitions of the Resource with type "rt" = "oic.r.crl".**

Property name	Value type	Mandatory	Access mode	Description
crldata	string	Yes	Read Write	Base64 BER encoded CRL data.
thisupdate	string	Yes	Read Write	UTC time of last CRL update.
n	multiple types: see schema	No	Read Write	
crlid	integer	Yes	Read Write	Local reference to a CRL Resource.
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 interface set supported by this Resource.
crldata	string		Read Write	Base64 BER encoded CRL data.



thisupdate	string		Read Write	UTC time of last CRL update.
crlid	integer		Read Write	Local reference to a CRL Resource.

7417 **C.7.6 CRUDN behaviour**

7418 Table C.13 defines the CRUDN operations that are supported on the "oic.r.crl" Resource Type.

7419 **Table C.13 – The CRUDN operations of the Resource with type "rt" = "oic.r.crl".**

Create	Read	Update	Delete	Notify
	get	post		observe

7420 **C.8 Certificate Signing Request**

7421 **C.8.1 Introduction**

7422 This Resource specifies a Certificate Signing Request.

7423 **C.8.2 Well-known URI**

7424 /oic/sec/csr

7425 **C.8.3 Resource type**

7426 The Resource Type is defined as: "oic.r.csr".

7427 **C.8.4 OpenAPI 2.0 definition**

```

7428 {
7429   "swagger": "2.0",
7430   "info": {
7431     "title": "Certificate Signing Request",
7432     "version": "v1.0-20150819",
7433     "license": {
7434       "name": "OCF Data Model License",
7435       "url":
7436         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7437         CENSE.md",
7438       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7439         reserved."
7440     },
7441     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7442   },
7443   "schemes": ["http"],
7444   "consumes": ["application/json"],
7445   "produces": ["application/json"],
7446   "paths": {
7447     "/oic/sec/csr" : {
7448       "get": {
7449         "description": "This Resource specifies a Certificate Signing Request.\n",
7450         "parameters": [
7451           { "$ref": "#/parameters/interface" }
7452         ],
7453         "responses": {
7454           "200": {
7455             "description": "",
7456             "x-example":
7457               {
7458                 "rt": ["oic.r.csr"],
7459                 "encoding": "oic.sec.encoding.pem",
7460                 "csr": "PEMENCODEDCSR"
7461               },
7462             "schema": { "$ref": "#/definitions/Csr" }
7463           },
7464           "404": {
7465             "description": "The Device does not support certificates and generating CSRs."
7466           }
7467         }
7468       }
7469     }
7470   }

```

```

7467         "503": {
7468             "description": "The Device is not yet ready to return a response. Try again later."
7469         }
7470     }
7471 }
7472 }
7473 },
7474 "parameters": {
7475     "interface": {
7476         "in": "query",
7477         "name": "if",
7478         "type": "string",
7479         "enum": ["oic.if.baseline"]
7480     }
7481 },
7482 "definitions": {
7483     "Csr": {
7484         "properties": {
7485             "rt": {
7486                 "description": "Resource Type of the Resource.",
7487                 "items": {
7488                     "maxLength": 64,
7489                     "type": "string",
7490                     "enum": ["oic.r.csr"]
7491                 },
7492                 "minItems": 1,
7493                 "readOnly": true,
7494                 "type": "array"
7495             },
7496             "encoding": {
7497                 "description": "A string specifying the encoding format of the data contained in CSR.",
7498                 "x-detail-desc": [
7499                     "oic.sec.encoding.pem - Encoding for PEM encoded CSR.",
7500                     "oic.sec.encoding.der - Encoding for DER encoded CSR."
7501                 ],
7502                 "enum": [
7503                     "oic.sec.encoding.pem",
7504                     "oic.sec.encoding.der"
7505                 ],
7506                 "readOnly": true,
7507                 "type": "string"
7508             },
7509             "n": {
7510                 "$ref":
7511                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7512                 schema.json#/definitions/n"
7513             },
7514             "id": {
7515                 "$ref":
7516                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7517                 schema.json#/definitions/id"
7518             },
7519             "csr": {
7520                 "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property.",
7521                 "maxLength": 3072,
7522                 "readOnly": true,
7523                 "type": "string"
7524             },
7525             "if": {
7526                 "description": "The interface set supported by this Resource.",
7527                 "items": {
7528                     "enum": [
7529                         "oic.if.baseline"
7530                     ],
7531                     "type": "string"
7532                 },
7533                 "minItems": 1,
7534                 "readOnly": true,
7535                 "type": "array"
7536             }
7537         }

```

```

7538     "type" : "object",
7539     "required": ["csr", "encoding"]
7540   }
7541 }
7542 }
7543

```

### 7544 C.8.5 Property definition

7545 Table C.14 defines the Properties that are part of the "oic.r.csr" Resource Type.

7546 **Table C.14 – The Property definitions of the Resource with type "rt" = "oic.r.csr".**

Property name	Value type	Mandatory	Access mode	Description
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in CSR.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property.

### 7547 C.8.6 CRUDN behaviour

7548 Table C.15 defines the CRUDN operations that are supported on the "oic.r.csr" Resource Type.

7549 **Table C.15 – The CRUDN operations of the Resource with type "rt" = "oic.r.csr".**

Create	Read	Update	Delete	Notify
	get			observe

## 7550 C.9 Device Owner Transfer Method

### 7551 C.9.1 Introduction

7552 This Resource specifies properties needed to establish a Device owner.

7553

### 7554 C.9.2 Well-known URI

7555 /oic/sec/doxm

### 7556 C.9.3 Resource type

7557 The Resource Type is defined as: "oic.r.doxm".

### 7558 C.9.4 OpenAPI 2.0 definition

```

7559 {
7560   "swagger": "2.0",
7561   "info": {

```

```

7562     "title": "Device Owner Transfer Method",
7563     "version": "v1.0-20181001",
7564     "license": {
7565         "name": "OCF Data Model License",
7566         "url":
7567         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7568 CENSE.md",
7569         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7570 reserved."
7571     },
7572     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7573 },
7574 "schemas": ["http"],
7575 "consumes": ["application/json"],
7576 "produces": ["application/json"],
7577 "paths": {
7578     "/oic/sec/doxm" : {
7579         "get": {
7580             "description": "This Resource specifies properties needed to establish a Device owner.\n",
7581             "parameters": [
7582                 {"$ref": "#/parameters/interface"}
7583             ],
7584             "responses": {
7585                 "200": {
7586                     "description": "",
7587                     "x-example":
7588                     {
7589                         "rt": ["oic.r.doxm"],
7590                         "oxms": [ 0, 2, 3 ],
7591                         "oxmsel": 0,
7592                         "sct": 16,
7593                         "owned": true,
7594                         "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7595                         "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7596                         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7597                     }
7598                 },
7599                 "schema": { "$ref": "#/definitions/Doxm" }
7600             },
7601             "400": {
7602                 "description": "The request is invalid."
7603             }
7604         }
7605     },
7606     "post": {
7607         "description": "Updates the DOXM Resource data.\n",
7608         "parameters": [
7609             {"$ref": "#/parameters/interface"},
7610             {
7611                 "name": "body",
7612                 "in": "body",
7613                 "required": true,
7614                 "schema": { "$ref": "#/definitions/Doxm-Update" },
7615                 "x-example":
7616                 {
7617                     "oxmsel": 0,
7618                     "owned": true,
7619                     "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7620                     "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7621                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7622                 }
7623             }
7624         ],
7625         "responses": {
7626             "400": {
7627                 "description": "The request is invalid."
7628             },
7629             "204": {
7630                 "description": "The DOXM entry is updated."
7631             }
7632         }
7633     }

```

```

7633     }
7634   }
7635 },
7636 "parameters": {
7637   "interface" : {
7638     "in" : "query",
7639     "name" : "if",
7640     "type" : "string",
7641     "enum" : ["oic.if.baseline"]
7642   }
7643 },
7644 "definitions": {
7645   "Doxm" : {
7646     "properties": {
7647       "rowneruuid": {
7648         "description": "Format pattern according to IETF RFC 4122.",
7649         "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}$",
7650         "type": "string"
7651       },
7652       "oxms": {
7653         "description": "List of supported owner transfer methods.",
7654         "items": {
7655           "description": "The Device owner transfer methods that may be selected at Device on-
7656 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
7657 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
7658 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
7659 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
7660 (deprecated).",
7661           "type": "integer"
7662         },
7663         "readOnly": true,
7664         "type": "array"
7665       },
7666       "devowneruuid": {
7667         "description": "Format pattern according to IETF RFC 4122.",
7668         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7669         "type": "string"
7670       },
7671       "deviceuuid": {
7672         "description": "The uuid formatted identity of the Device\nFormat pattern according to
7673 IETF RFC 4122.",
7674         "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}$",
7675         "type": "string"
7676       },
7677       "owned": {
7678         "description": "Ownership status flag.",
7679         "type": "boolean"
7680       },
7681       "n": {
7682         "$ref":
7683 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7684 schema.json#/definitions/n"
7685       },
7686       "id": {
7687         "$ref":
7688 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7689 schema.json#/definitions/id"
7690       },
7691       "oxmsel": {
7692         "description": "The selected owner transfer method used during on-boarding\nThe Device
7693 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7694 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7695 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7696 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7697 method (oic.sec.doxm.dcap) (deprecated).",
7698         "type": "integer"
7699       },
7700       "sct": {
7701         "description": "Bitmask encoding of supported credential types\nCredential Types -
7702 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7703 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or

```

```

7704 password32 - Asymmetric encryption key.",
7705     "maximum": 63,
7706     "minimum": 0,
7707     "type": "integer",
7708     "readOnly": true
7709 },
7710 "rt" : {
7711     "description": "Resource Type of the Resource.",
7712     "items": {
7713         "maxLength": 64,
7714         "type": "string",
7715         "enum": ["oic.r.doxm"]
7716     },
7717     "minItems": 1,
7718     "readOnly": true,
7719     "type": "array"
7720 },
7721 "if": {
7722     "description": "The interface set supported by this Resource.",
7723     "items": {
7724         "enum": [
7725             "oic.if.baseline"
7726         ],
7727         "type": "string"
7728     },
7729     "minItems": 1,
7730     "readOnly": true,
7731     "type": "array"
7732 }
7733 },
7734 "type" : "object",
7735 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
7736 },
7737 "Doxm-Update" : {
7738     "properties": {
7739         "rowneruuid": {
7740             "description": "Format pattern according to IETF RFC 4122.",
7741             "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}$",
7742             "type": "string"
7743         },
7744         "devowneruuid": {
7745             "description": "Format pattern according to IETF RFC 4122.",
7746             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7747             "type": "string"
7748         },
7749         "deviceuuid": {
7750             "description": "The uuid formatted identity of the Device\nFormat pattern according to
7751 IETF RFC 4122.",
7752             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7753 9]{12}$",
7754             "type": "string"
7755         },
7756         "owned": {
7757             "description": "Ownership status flag.",
7758             "type": "boolean"
7759         },
7760         "oxmsel": {
7761             "description": "The selected owner transfer method used during on-boarding\nThe Device
7762 owner transfer methods that may be selected at Device on-boarding. Each value indicates a specific
7763 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7764 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7765 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7766 method (oic.sec.doxm.dcap) (deprecated).",
7767             "type": "integer"
7768         }
7769     },
7770 "type" : "object"
7771 }
7772 }
7773 }
7774 }

```

7775 **C.9.5 Property definition**

7776 Table C.16 defines the Properties that are part of the "oic.r.doxm" Resource Type.

7777 **Table C.16 – The Property definitions of the Resource with type "rt" = "oic.r.doxm".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
owned	boolean	Yes	Read Write	Ownership status flag.
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).
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the Device 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.
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
n	multiple types: see schema	No	Read Write	
oxms	array: see schema	Yes	Read Only	List of supported owner transfer methods.
devowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
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.
rowneruuid	string		Read Write	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.
deviceuuid	string		Read Write	The uuid formatted identity of the Device Format pattern according to IETF RFC 4122.



7778 **C.9.6 CRUDN behaviour**

7779 Table C.17 defines the CRUDN operations that are supported on the "oic.r.doxxm" Resource Type.

7780 **Table C.17 – The CRUDN operations of the Resource with type "rt" = "oic.r.doxxm".**

Create	Read	Update	Delete	Notify
	get	post		observe

7781 **C.10 Device Provisioning Status**

7782 **C.10.1 Introduction**

7783 This Resource specifies Device provisioning status.

7784

7785 **C.10.2 Well-known URI**

7786 /oic/sec/pstat

7787 **C.10.3 Resource type**

7788 The Resource Type is defined as: "oic.r.pstat".

7789 **C.10.4 OpenAPI 2.0 definition**

```

7790 {
7791   "swagger": "2.0",
7792   "info": {
7793     "title": "Device Provisioning Status",
7794     "version": "v1.0-20191001",
7795     "license": {
7796       "name": "OCF Data Model License",
7797       "url":
7798         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7799         CENSE.md",
7800       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7801         reserved."
7802     },
7803     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7804   },
7805   "schemes": ["http"],
7806   "consumes": ["application/json"],
7807   "produces": ["application/json"],
7808   "paths": {
7809     "/oic/sec/pstat" : {
7810       "get": {
7811         "description": "This Resource specifies Device provisioning status.\n",
7812         "parameters": [
7813           { "$ref": "#/parameters/interface" }
7814         ],
7815         "responses": {
7816           "200": {
7817             "description": "",
7818             "x-example":
7819               {
7820                 "rt": ["oic.r.pstat"],
7821                 "dos": {"s": 3, "p": true},
7822                 "isop": true,
7823                 "cm": 8,
7824                 "tm": 60,
7825                 "om": 2,
7826                 "sm": 7,
7827                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7828               },
7829             "schema": { "$ref": "#/definitions/Pstat" }
7830           },
7831           "400": {
7832             "description": "The request is invalid."
7833           }
7834         }
7835       }
7836     }
7837   }

```

```

7835     },
7836     "post": {
7837         "description": "Sets or updates Device provisioning status data.\n",
7838         "parameters": [
7839             { "$ref": "#/parameters/interface" },
7840             {
7841                 "name": "body",
7842                 "in": "body",
7843                 "required": true,
7844                 "schema": { "$ref": "#/definitions/Pstat-Update" },
7845                 "x-example":
7846                 {
7847                     "dos": { "s": 3 },
7848                     "tm": 60,
7849                     "om": 2,
7850                     "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7851                 }
7852             }
7853         ],
7854         "responses": {
7855             "400": {
7856                 "description": "The request is invalid."
7857             },
7858             "204": {
7859                 "description": "The PSTAT entry is updated."
7860             }
7861         }
7862     }
7863 },
7864 },
7865 "parameters": {
7866     "interface": {
7867         "in": "query",
7868         "name": "if",
7869         "type": "string",
7870         "enum": ["oic.if.baseline"]
7871     }
7872 },
7873 "definitions": {
7874     "Pstat": {
7875         "properties": {
7876             "rowneruuid": {
7877                 "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7878 according to IETF RFC 4122.",
7879                 "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}$",
7880                 "type": "string"
7881             },
7882             "rt": {
7883                 "description": "Resource Type of the Resource.",
7884                 "items": {
7885                     "maxLength": 64,
7886                     "type": "string",
7887                     "enum": ["oic.r.pstat"]
7888                 },
7889                 "minItems": 1,
7890                 "readOnly": true,
7891                 "type": "array"
7892             },
7893             "om": {
7894                 "description": "Current operational mode\nDevice provisioning operation may be server
7895 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7896 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7897 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7898 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7899                 "maximum": 7,
7900                 "minimum": 1,
7901                 "type": "integer"
7902             },
7903             "cm": {
7904                 "description": "Current Device provisioning mode\nDevice provisioning mode maintains a
7905 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character

```

```

7906 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7907 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7908 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7909 Software Version Validation128 - Initiate Secure Software Update.",
7910     "maximum": 255,
7911     "minimum": 0,
7912     "type": "integer",
7913     "readOnly": true
7914 },
7915 "n": {
7916     "$ref":
7917 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7918 schema.json#/definitions/n"
7919 },
7920 "id": {
7921     "$ref":
7922 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7923 schema.json#/definitions/id"
7924 },
7925 "isop": {
7926     "description": "true indicates Device is operational.",
7927     "readOnly": true,
7928     "type": "boolean"
7929 },
7930 "tm": {
7931     "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
7932 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
7933 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7934 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7935 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7936 Software Version Validation128 - Initiate Secure Software Update.",
7937     "maximum": 255,
7938     "minimum": 0,
7939     "type": "integer"
7940 },
7941 "sm": {
7942     "description": "Supported operational modes\nDevice provisioning operation may be server
7943 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7944 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7945 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7946 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
7947     "maximum": 7,
7948     "minimum": 1,
7949     "type": "integer",
7950     "readOnly": true
7951 },
7952 "dos": {
7953     "description": "Device on-boarding state\nDevice operation state machine.",
7954     "properties": {
7955         "p": {
7956             "default": true,
7957             "description": "'p' is TRUE when the 's' state is pending until all necessary changes
7958 to Device Resources are complete.",
7959             "readOnly": true,
7960             "type": "boolean"
7961         },
7962         "s": {
7963             "description": "The current or pending operational state.",
7964             "x-detail-desc": [
7965                 "0 - RESET - Device reset state.",
7966                 "1 - RFOTM - Ready for Device owner transfer method state.",
7967                 "2 - RFPRO - Ready for Device provisioning state.",
7968                 "3 - RFNOP - Ready for Device normal operation state.",
7969                 "4 - SRESET - The Device is in a soft reset state."
7970             ],
7971             "maximum": 4,
7972             "minimum": 0,
7973             "type": "integer"
7974         }
7975     },
7976     "required": [

```

```

7977         "s"
7978     ],
7979     "type": "object"
7980 },
7981 "if" : {
7982     "description": "The interface set supported by this Resource.",
7983     "items": {
7984         "enum": [
7985             "oic.if.baseline"
7986         ],
7987         "type": "string"
7988     },
7989     "minItems": 1,
7990     "readOnly": true,
7991     "type": "array"
7992 }
7993 },
7994 "type" : "object",
7995 "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
7996 },
7997 "Pstat-Update" : {
7998     "properties": {
7999         "rowneruuid": {
8000             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
8001 according to IETF RFC 4122.",
8002             "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}$",
8003             "type": "string"
8004         },
8005         "om": {
8006             "description": "Current operational mode\nDevice provisioning operation may be server
8007 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
8008 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
8009 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
8010 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.",
8011             "maximum": 7,
8012             "minimum": 1,
8013             "type": "integer"
8014         },
8015         "tm": {
8016             "description": "Target Device provisioning mode\nDevice provisioning mode maintains a
8017 bitmask of the possible provisioning states of a Device. The value can be either 8 or 16 character
8018 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
8019 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
8020 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
8021 Software Version Validation128 - Initiate Secure Software Update.",
8022             "maximum": 255,
8023             "minimum": 0,
8024             "type": "integer"
8025         },
8026         "dos": {
8027             "description": "Device on-boarding state\nDevice operation state machine.",
8028             "properties": {
8029                 "p": {
8030                     "default": true,
8031                     "description": "'p' is TRUE when the 's' state is pending until all necessary changes
8032 to Device Resources are complete.",
8033                     "readOnly": true,
8034                     "type": "boolean"
8035                 },
8036                 "s": {
8037                     "description": "The current or pending operational state.",
8038                     "x-detail-desc": [
8039                         "0 - RESET - Device reset state.",
8040                         "1 - RFOTM - Ready for Device owner transfer method state.",
8041                         "2 - RFPPO - Ready for Device provisioning state.",
8042                         "3 - RFNOP - Ready for Device normal operation state.",
8043                         "4 - SRESET - The Device is in a soft reset state."
8044                     ],
8045                     "maximum": 4,
8046                     "minimum": 0,
8047                     "type": "integer"

```

```

8048     }
8049     },
8050     "required": [
8051         "s"
8052     ],
8053     "type": "object"
8054 }
8055 },
8056 "type" : "object"
8057 }
8058 }
8059 }
8060 }

```

8061 **C.10.5 Property definition**

8062 Table C.18 defines the Properties that are part of the "oic.r.pstat" Resource Type.

8063 **Table C.18 – The Property definitions of the Resource with type "rt" = "oic.r.pstat".**

Property name	Value type	Mandatory	Access mode	Description
dos	object: see schema	No	Read Write	Device on-boarding state machine. Device operation state machine.
rowneruuid	string	No	Read Write	The UUID formatted identity of the Resource owner. Format pattern according to IETF RFC 4122.
tm	integer	No	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 it's 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 ACLs 64 - Initiate Software Version Validation 128 - Initiate Secure Software Update.
om	integer	No	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 modes 1 - Server-directed utilizing multiple provisioning services 2 - Server-directed utilizing a single provisioning service 4 - Client-directed provisioning 8 - Unused 16 - Unused 32 - Unused 64 - Unused 128 - Unused.
isop	boolean	Yes	Read Only	true indicates Device is operational.
cm	integer	Yes	Read Only	Current Device provisioning mode Device provisioning mode maintains a bitmask of the

				<p>possible provisioning states of a Device. The value can be either 8 or 16 character in length. If it's 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>
sm	integer	Yes	Read Only	<p>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 -</p>

				Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused.
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.
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 it's 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.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The interface set supported by this Resource.
dos	object: see schema	Yes	Read Write	Device on-boarding state Device operation state machine.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.

8064 **C.10.6 CRUDN behaviour**

8065 Table C.19 defines the CRUDN operations that are supported on the "oic.r.pstat" Resource Type.

8066 **Table C.19 – The CRUDN operations of the Resource with type "rt" = "oic.r.pstat".**

Create	Read	Update	Delete	Notify
--------	------	--------	--------	--------

	get	post		observe
--	-----	------	--	---------

## 8067 C.11 Asserted Roles

### 8068 C.11.1 Introduction

8069 This Resource specifies roles that have been asserted.

8070

### 8071 C.11.2 Well-known URI

8072 /oic/sec/roles

### 8073 C.11.3 Resource type

8074 The Resource Type is defined as: "oic.r.roles".

### 8075 C.11.4 OpenAPI 2.0 definition

```

8076 {
8077   "swagger": "2.0",
8078   "info": {
8079     "title": "Asserted Roles",
8080     "version": "v1.0-20170323",
8081     "license": {
8082       "name": "OCF Data Model License",
8083       "url":
8084         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8085         CENSE.md",
8086       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8087         reserved."
8088     },
8089     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8090   },
8091   "schemes": ["http"],
8092   "consumes": ["application/json"],
8093   "produces": ["application/json"],
8094   "paths": {
8095     "/oic/sec/roles" : {
8096       "get": {
8097         "description": "This Resource specifies roles that have been asserted.\n",
8098         "parameters": [
8099           {"$ref": "#/parameters/interface"}
8100         ],
8101         "responses": {
8102           "200": {
8103             "description": "",
8104             "x-example":
8105               {
8106                 "roles" :[
8107                   {
8108                     "credid":1,
8109                     "credtype":8,
8110                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
8111                     "publicdata":
8112                       {
8113                         "encoding":"oic.sec.encoding.pem",
8114                         "data":"PEMENCODEDROLECERT"
8115                       },
8116                     "optionaldata":
8117                       {
8118                         "revstat": false,
8119                         "encoding":"oic.sec.encoding.pem",
8120                         "data":"PEMENCODEDISSUERCERT"
8121                       }
8122                   },
8123                   {
8124                     "credid":2,
8125                     "credtype":8,
8126                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
8127                     "publicdata":

```

```

8128         {
8129             "encoding": "oic.sec.encoding.pem",
8130             "data": "PEMENCODEDROLECERT"
8131         },
8132         "optionaldata":
8133         {
8134             "revstat": false,
8135             "encoding": "oic.sec.encoding.pem",
8136             "data": "PEMENCODEDISSUERCERT"
8137         }
8138     },
8139 ],
8140     "rt": ["oic.r.roles"],
8141     "if": ["oic.if.baseline"]
8142 }
8143 ,
8144     "schema": { "$ref": "#/definitions/Roles" }
8145 },
8146     "400": {
8147         "description": "The request is invalid."
8148     }
8149 }
8150 },
8151     "post": {
8152         "description": "Update the roles Resource, i.e., assert new roles to this server.\n\nNew
8153 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
8154 same) are not added to the Resource (and 204 is\nreturned).\n\nThe provided credid values are
8155 ignored, the Resource assigns its own.\n",
8156         "parameters": [
8157             { "$ref": "#/parameters/interface" },
8158             {
8159                 "name": "body",
8160                 "in": "body",
8161                 "required": true,
8162                 "schema": { "$ref": "#/definitions/Roles-update" },
8163                 "x-example":
8164                 {
8165                     "roles" :[
8166                         {
8167                             "credid":1,
8168                             "credtype":8,
8169                             "subjectuuid":"00000000-0000-0000-0000-000000000000",
8170                             "publicdata":
8171                             {
8172                                 "encoding": "oic.sec.encoding.pem",
8173                                 "data": "PEMENCODEDROLECERT"
8174                             },
8175                             "optionaldata":
8176                             {
8177                                 "revstat": false,
8178                                 "encoding": "oic.sec.encoding.pem",
8179                                 "data": "PEMENCODEDISSUERCERT"
8180                             }
8181                         },
8182                         {
8183                             "credid":2,
8184                             "credtype":8,
8185                             "subjectuuid":"00000000-0000-0000-0000-000000000000",
8186                             "publicdata":
8187                             {
8188                                 "encoding": "oic.sec.encoding.pem",
8189                                 "data": "PEMENCODEDROLECERT"
8190                             },
8191                             "optionaldata":
8192                             {
8193                                 "revstat": false,
8194                                 "encoding": "oic.sec.encoding.pem",
8195                                 "data": "PEMENCODEDISSUERCERT"
8196                             }
8197                         }
8198                     ]
8199                 }
8200             }
8201         ]
8202     }
8203 }

```

```

8199     }
8200   }
8201 ],
8202 "responses": {
8203   "400": {
8204     "description": "The request is invalid."
8205   },
8206   "204": {
8207     "description": "The roles entry is updated."
8208   }
8209 }
8210 },
8211 "delete": {
8212   "description": "Deletes roles Resource entries.\nWhen DELETE is used without query
8213 parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
8214 entries matching\nthe query parameter are deleted.\n",
8215   "parameters": [
8216     { "$ref": "#/parameters/interface" },
8217     { "$ref": "#/parameters/roles-filtered" }
8218   ],
8219   "responses": {
8220     "200": {
8221       "description": "The specified or all roles Resource entries have been successfully
8222 deleted."
8223     },
8224     "400": {
8225       "description": "The request is invalid."
8226     }
8227   }
8228 }
8229 },
8230 },
8231 "parameters": {
8232   "interface": {
8233     "in": "query",
8234     "name": "if",
8235     "type": "string",
8236     "enum": ["oic.if.baseline"]
8237   },
8238   "roles-filtered": {
8239     "in": "query",
8240     "name": "credid",
8241     "required": false,
8242     "type": "integer",
8243     "description": "Only applies to the credential with the specified credid.",
8244     "x-example": 2112
8245   }
8246 },
8247 "definitions": {
8248   "Roles": {
8249     "properties": {
8250       "rt": {
8251         "description": "Resource Type of the Resource.",
8252         "items": {
8253           "maxLength": 64,
8254           "type": "string",
8255           "enum": ["oic.r.roles"]
8256         },
8257         "minItems": 1,
8258         "readOnly": true,
8259         "type": "array"
8260       },
8261       "n": {
8262         "$ref":
8263 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8264 schema.json#/definitions/n"
8265       },
8266       "id": {
8267         "$ref":
8268 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8269 schema.json#/definitions/id"

```

```

8270     },
8271     "roles": {
8272         "description": "List of role certificates.",
8273         "items": {
8274             "properties": {
8275                 "credid": {
8276                     "description": "Local reference to a credential Resource.",
8277                     "type": "integer"
8278                 },
8279                 "credtype": {
8280                     "description": "Representation of this credential's type\nCredential Types - Cred
8281 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8282 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
8283 password32 - Asymmetric encryption key.",
8284                     "maximum": 63,
8285                     "minimum": 0,
8286                     "type": "integer"
8287                 },
8288                 "credusage": {
8289                     "description": "A string that provides hints about how/where the cred is used\nThe
8290 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8291 Certificateoic.sec.cred.rolcert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8292 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
8293                     "enum": [
8294                         "oic.sec.cred.trustca",
8295                         "oic.sec.cred.cert",
8296                         "oic.sec.cred.rolcert",
8297                         "oic.sec.cred.mfgtrustca",
8298                         "oic.sec.cred.mfgcert"
8299                     ],
8300                     "type": "string"
8301                 },
8302                 "crms": {
8303                     "description": "The refresh methods that may be used to update this credential.",
8304                     "items": {
8305                         "description": "Each enum represents a method by which the credentials are
8306 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8307 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8308 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8309 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
8310                         "enum": [
8311                             "oic.sec.crm.pro",
8312                             "oic.sec.crm.psk",
8313                             "oic.sec.crm.rdp",
8314                             "oic.sec.crm.skdc",
8315                             "oic.sec.crm.pk10"
8316                         ],
8317                         "type": "string"
8318                     },
8319                     "type": "array"
8320                 },
8321                 "optionaldata": {
8322                     "description": "Credential revocation status information\nOptional credential
8323 contents describes revocation status for this credential.",
8324                     "properties": {
8325                         "data": {
8326                             "description": "This is the encoded structure.",
8327                             "type": "string"
8328                         },
8329                         "encoding": {
8330                             "description": "A string specifying the encoding format of the data contained in
8331 the optdata.",
8332                             "x-detail-desc": [
8333                                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8334                                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8335                                 "oic.sec.encoding.base64 - Base64 encoded object.",
8336                                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8337                                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8338                                 "oic.sec.encoding.raw - Raw hex encoded data."
8339                             ],
8340                             "type": "string"
8341                         }
8342                     }
8343                 }
8344             }
8345         }
8346     }

```

```

8341         "oic.sec.encoding.jwt",
8342         "oic.sec.encoding.cwt",
8343         "oic.sec.encoding.base64",
8344         "oic.sec.encoding.pem",
8345         "oic.sec.encoding.der",
8346         "oic.sec.encoding.raw"
8347     ],
8348     "type": "string"
8349 },
8350 "revstat": {
8351     "description": "Revocation status flag - true = revoked.",
8352     "type": "boolean"
8353 }
8354 },
8355 "required": [
8356     "revstat"
8357 ],
8358 "type": "object"
8359 },
8360 "period": {
8361     "description": "String with RFC5545 Period.",
8362     "type": "string"
8363 },
8364 "privatedata": {
8365     "description": "Private credential information\nCredential Resource non-public
8366 contents.",
8367     "properties": {
8368         "data": {
8369             "description": "The encoded value.",
8370             "maxLength": 3072,
8371             "type": "string"
8372         },
8373         "encoding": {
8374             "description": "A string specifying the encoding format of the data contained in
8375 the privdata.",
8376             "x-detail-desc": [
8377                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8378                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8379                 "oic.sec.encoding.base64 - Base64 encoded object.",
8380                 "oic.sec.encoding.uri - URI reference.",
8381                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
8382 referenced using a handle.",
8383                 "oic.sec.encoding.raw - Raw hex encoded data."
8384             ],
8385             "enum": [
8386                 "oic.sec.encoding.jwt",
8387                 "oic.sec.encoding.cwt",
8388                 "oic.sec.encoding.base64",
8389                 "oic.sec.encoding.uri",
8390                 "oic.sec.encoding.handle",
8391                 "oic.sec.encoding.raw"
8392             ],
8393             "type": "string"
8394         },
8395         "handle": {
8396             "description": "Handle to a key storage Resource.",
8397             "type": "integer"
8398         }
8399     },
8400     "required": [
8401         "encoding"
8402     ],
8403     "type": "object"
8404 },
8405 "publicdata": {
8406     "description": "Public credential information.",
8407     "properties": {
8408         "data": {
8409             "description": "This is the encoded value.",
8410             "maxLength": 3072,
8411             "type": "string"

```

```

8412     },
8413     "encoding": {
8414         "description": "A string specifying the encoding format of the data contained in
8415 the pubdata.",
8416         "x-detail-desc": [
8417             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8418             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8419             "oic.sec.encoding.base64 - Base64 encoded object.",
8420             "oic.sec.encoding.uri - URI reference.",
8421             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8422             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8423             "oic.sec.encoding.raw - Raw hex encoded data."
8424         ],
8425         "enum": [
8426             "oic.sec.encoding.jwt",
8427             "oic.sec.encoding.cwt",
8428             "oic.sec.encoding.base64",
8429             "oic.sec.encoding.uri",
8430             "oic.sec.encoding.pem",
8431             "oic.sec.encoding.der",
8432             "oic.sec.encoding.raw"
8433         ],
8434         "type": "string"
8435     }
8436 },
8437 "type": "object"
8438 },
8439 "roleid": {
8440     "description": "The role this credential possesses\nSecurity role specified as an
8441 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8442     "properties": {
8443         "authority": {
8444             "description": "The Authority component of the entity being identified. A NULL
8445 <Authority> refers to the local entity or Device.",
8446             "type": "string"
8447         },
8448         "role": {
8449             "description": "The ID of the role being identified.",
8450             "type": "string"
8451         }
8452     },
8453     "required": [
8454         "role"
8455     ],
8456     "type": "object"
8457 },
8458 "subjectuuid": {
8459     "anyOf": [
8460         {
8461             "description": "The id of the Device, which the cred entry applies to or \"*\n
8462 for wildcard identity.",
8463             "pattern": "^\\*$",
8464             "type": "string"
8465         },
8466         {
8467             "description": "Format pattern according to IETF RFC 4122.",
8468             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8469 F0-9]{12}$",
8470             "type": "string"
8471         }
8472     ]
8473 }
8474 },
8475 "type": "object"
8476 },
8477 "type": "array"
8478 },
8479 "if": {
8480     "description": "The interface set supported by this Resource.",
8481     "items": {
8482         "enum": [

```

```

8483         "oic.if.baseline"
8484     ],
8485     "type": "string"
8486 },
8487     "minItems": 1,
8488     "readOnly": true,
8489     "type": "array"
8490 }
8491 },
8492     "type" : "object",
8493     "required": ["roles"]
8494 },
8495     "Roles-update" : {
8496         "properties": {
8497             "roles": {
8498                 "description": "List of role certificates.",
8499                 "items": {
8500                     "properties": {
8501                         "credid": {
8502                             "description": "Local reference to a credential Resource.",
8503                             "type": "integer"
8504                     },
8505                     "credtype": {
8506                         "description": "Representation of this credential's type\nCredential Types - Cred
8507 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8508 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
8509 password32 - Asymmetric encryption key.",
8510                         "maximum": 63,
8511                         "minimum": 0,
8512                         "type": "integer"
8513                     },
8514                     "credusage": {
8515                         "description": "A string that provides hints about how/where the cred is used\nThe
8516 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8517 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8518 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate.",
8519                         "enum": [
8520                             "oic.sec.cred.trustca",
8521                             "oic.sec.cred.cert",
8522                             "oic.sec.cred.rolecert",
8523                             "oic.sec.cred.mfgtrustca",
8524                             "oic.sec.cred.mfgcert"
8525                         ],
8526                         "type": "string"
8527                     },
8528                     "crms": {
8529                         "description": "The refresh methods that may be used to update this credential.",
8530                         "items": {
8531                             "description": "Each enum represents a method by which the credentials are
8532 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8533 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8534 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8535 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA.",
8536                             "enum": [
8537                                 "oic.sec.crm.pro",
8538                                 "oic.sec.crm.psk",
8539                                 "oic.sec.crm.rdp",
8540                                 "oic.sec.crm.skdc",
8541                                 "oic.sec.crm.pk10"
8542                             ],
8543                             "type": "string"
8544                         },
8545                         "type": "array"
8546                     },
8547                     "optionaldata": {
8548                         "description": "Credential revocation status information\nOptional credential
8549 contents describes revocation status for this credential.",
8550                         "properties": {
8551                             "data": {
8552                                 "description": "This is the encoded structure.",
8553                                 "type": "string"

```



```

8554     },
8555     "encoding": {
8556         "description": "A string specifying the encoding format of the data contained in
8557 the optdata.",
8558         "x-detail-desc": [
8559             "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8560             "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8561             "oic.sec.encoding.base64 - Base64 encoded object.",
8562             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8563             "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8564             "oic.sec.encoding.raw - Raw hex encoded data."
8565         ],
8566         "enum": [
8567             "oic.sec.encoding.jwt",
8568             "oic.sec.encoding.cwt",
8569             "oic.sec.encoding.base64",
8570             "oic.sec.encoding.pem",
8571             "oic.sec.encoding.der",
8572             "oic.sec.encoding.raw"
8573         ],
8574         "type": "string"
8575     },
8576     "revstat": {
8577         "description": "Revocation status flag - true = revoked.",
8578         "type": "boolean"
8579     }
8580 },
8581 "required": [
8582     "revstat"
8583 ],
8584 "type": "object"
8585 },
8586 "period": {
8587     "description": "String with RFC5545 Period.",
8588     "type": "string"
8589 },
8590 "privatedata": {
8591     "description": "Private credential information\nCredential Resource non-public
8592 contents.",
8593     "properties": {
8594         "data": {
8595             "description": "The encoded value.",
8596             "maxLength": 3072,
8597             "type": "string"
8598         },
8599         "encoding": {
8600             "description": "A string specifying the encoding format of the data contained in
8601 the privdata.",
8602             "x-detail-desc": [
8603                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8604                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8605                 "oic.sec.encoding.base64 - Base64 encoded object.",
8606                 "oic.sec.encoding.uri - URI reference.",
8607                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
8608 referenced using a handle.",
8609                 "oic.sec.encoding.raw - Raw hex encoded data."
8610             ],
8611             "enum": [
8612                 "oic.sec.encoding.jwt",
8613                 "oic.sec.encoding.cwt",
8614                 "oic.sec.encoding.base64",
8615                 "oic.sec.encoding.uri",
8616                 "oic.sec.encoding.handle",
8617                 "oic.sec.encoding.raw"
8618             ],
8619             "type": "string"
8620         },
8621         "handle": {
8622             "description": "Handle to a key storage Resource.",
8623             "type": "integer"
8624         }
8625     }

```

```

8625     },
8626     "required": [
8627         "encoding"
8628     ],
8629     "type": "object"
8630 },
8631 "publicdata": {
8632     "description": "Public credential information.",
8633     "properties": {
8634         "data": {
8635             "description": "The encoded value.",
8636             "maxLength": 3072,
8637             "type": "string"
8638         },
8639         "encoding": {
8640             "description": "A string specifying the encoding format of the data contained in
8641 the pubdata.",
8642             "x-detail-desc": [
8643                 "oic.sec.encoding.jwt - RFC7517 JSON web token (JWT) encoding.",
8644                 "oic.sec.encoding.cwt - RFC CBOR web token (CWT) encoding.",
8645                 "oic.sec.encoding.base64 - Base64 encoded object.",
8646                 "oic.sec.encoding.uri - URI reference.",
8647                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain.",
8648                 "oic.sec.encoding.der - Encoding for DER encoded certificate.",
8649                 "oic.sec.encoding.raw - Raw hex encoded data."
8650             ],
8651             "enum": [
8652                 "oic.sec.encoding.jwt",
8653                 "oic.sec.encoding.cwt",
8654                 "oic.sec.encoding.base64",
8655                 "oic.sec.encoding.uri",
8656                 "oic.sec.encoding.pem",
8657                 "oic.sec.encoding.der",
8658                 "oic.sec.encoding.raw"
8659             ],
8660             "type": "string"
8661         }
8662     },
8663     "type": "object"
8664 },
8665 "roleid": {
8666     "description": "The role this credential possesses\nSecurity role specified as an
8667 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or Device.",
8668     "properties": {
8669         "authority": {
8670             "description": "The Authority component of the entity being identified. A NULL
8671 <Authority> refers to the local entity or Device.",
8672             "type": "string"
8673         },
8674         "role": {
8675             "description": "The ID of the role being identified.",
8676             "type": "string"
8677         }
8678     },
8679     "required": [
8680         "role"
8681     ],
8682     "type": "object"
8683 },
8684 "subjectuuid": {
8685     "anyOf": [
8686         {
8687             "description": "The id of the Device, which the cred entry applies to or \"*\n
8688 for wildcard identity.",
8689             "pattern": "^\\*$",
8690             "type": "string"
8691         },
8692         {
8693             "description": "Format pattern according to IETF RFC 4122.",
8694             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8695 F0-9]{12}$",

```

```

8696         "type": "string"
8697     }
8698   ]
8699 }
8700 },
8701 "type": "object"
8702 },
8703 "type": "array"
8704 }
8705 },
8706 "type" : "object",
8707 "required": ["roles"]
8708 }
8709 }
8710 }
8711

```

### 8712 C.11.5 Property definition

8713 Table C.20 defines the Properties that are part of the "oic.r.roles" Resource Type.

8714 **Table C.20 – 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	
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 interface set supported by this Resource.
roles	array: see schema	Yes	Read Write	List of role certificates.

### 8715 C.11.6 CRUDN behaviour

8716 Table C.21 defines the CRUDN operations that are supported on the "oic.r.roles" Resource Type.

8717 **Table C.21 – The CRUDN operations of the Resource with type "rt" = "oic.r.roles".**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

## 8718 C.12 Signed Access Control List

### 8719 C.12.1 Introduction

8720 This Resource specifies a signed ACL object.

8721

### 8722 C.12.2 Well-known URI

8723 /oic/sec/sacl

### 8724 C.12.3 Resource type

8725 The Resource Type is defined as: "oic.r.sacl".

### 8726 C.12.4 OpenAPI 2.0 definition

```

8727 {
8728   "swagger": "2.0",
8729   "info": {

```

```

8730     "title": "Signed Access Control List",
8731     "version": "v1.0-20150819",
8732     "license": {
8733         "name": "OCF Data Model License",
8734         "url":
8735     "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8736     CENSE.md",
8737         "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8738     reserved."
8739     },
8740     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8741 },
8742 "schemes": ["http"],
8743 "consumes": ["application/json"],
8744 "produces": ["application/json"],
8745 "paths": {
8746     "/oic/sec/sacl" : {
8747         "get": {
8748             "description": "This Resource specifies a signed ACL object.\n",
8749             "parameters": [
8750                 {"$ref": "#/parameters/interface"}
8751             ],
8752             "responses": {
8753                 "200": {
8754                     "description": "",
8755                     "x-example":
8756                 {
8757                     "rt": ["oic.r.sacl"],
8758                     "aclist2": [
8759                         {
8760                             "aceid": 1,
8761                             "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8762                             "resources": [
8763                                 {
8764                                     "href": "/temp",
8765                                     "rt": ["oic.r.temperature"],
8766                                     "if": ["oic.if.baseline", "oic.if.a"]
8767                                 },
8768                                 {
8769                                     "href": "/temp",
8770                                     "rt": ["oic.r.temperature"],
8771                                     "if": ["oic.if.baseline", "oic.if.s"]
8772                                 }
8773                             ],
8774                             "permission": 31,
8775                             "validity": [
8776                                 {
8777                                     "period": "20160101T180000Z/20170102T070000Z",
8778                                     "recurrence": [ "DSTART:XXXXX",
8779     "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8780                                 },
8781                                 {
8782                                     "period": "20160101T180000Z/PT5H30M",
8783                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8784                                 }
8785                             ]
8786                         },
8787                     ],
8788                     "aceid": 2,
8789                     "subject": {
8790                         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8791                         "role": "SOME_STRING"
8792                     },
8793                     "resources": [
8794                         {
8795                             "href": "/light",
8796                             "rt": ["oic.r.light"],
8797                             "if": ["oic.if.baseline", "oic.if.a"]
8798                         },
8799                         {
8800                             "href": "/door",

```

```

8801         "rt": ["oic.r.door"],
8802         "if": ["oic.if.baseline", "oic.if.a"]
8803     }
8804 ],
8805     "permission": 15
8806 }
8807 ],
8808     "signature": {
8809         "sigtype": "oic.sec.sigtype.pk7",
8810         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8811     }
8812 },
8813     "schema": { "$ref": "#/definitions/Sacl" }
8814 }
8815 },
8816 },
8817 "post": {
8818     "description": "Sets the sacl Resource data.\n",
8819     "parameters": [
8820         { "$ref": "#/parameters/interface" },
8821         {
8822             "name": "body",
8823             "in": "body",
8824             "required": true,
8825             "schema": { "$ref": "#/definitions/Sacl" },
8826             "x-example":
8827                 {
8828                     "acllist2": [
8829                         {
8830                             "aceid": 1,
8831                             "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8832                             "resources": [
8833                                 {
8834                                     "href": "/temp",
8835                                     "rt": ["oic.r.temperature"],
8836                                     "if": ["oic.if.baseline", "oic.if.a"]
8837                                 },
8838                                 {
8839                                     "href": "/temp",
8840                                     "rt": ["oic.r.temperature"],
8841                                     "if": ["oic.if.baseline", "oic.if.s"]
8842                                 }
8843                             ],
8844                             "permission": 31,
8845                             "validity": [
8846                                 {
8847                                     "period": "20160101T180000Z/20170102T070000Z",
8848                                     "recurrence": [ "DSTART:XXXXX",
8849 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8850                                 },
8851                                 {
8852                                     "period": "20160101T180000Z/PT5H30M",
8853                                     "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8854                                 }
8855                             ]
8856                         },
8857                     {
8858                         "aceid": 2,
8859                         "subject": {
8860                             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8861                             "role": "SOME_STRING"
8862                         },
8863                         "resources": [
8864                             {
8865                                 "href": "/light",
8866                                 "rt": ["oic.r.light"],
8867                                 "if": ["oic.if.baseline", "oic.if.a"]
8868                             },
8869                             {
8870                                 "href": "/door",
8871                                 "rt": ["oic.r.door"],

```

```

8872         "if": ["oic.if.baseline", "oic.if.a"]
8873     }
8874 ],
8875     "permission": 15
8876 }
8877 ],
8878     "signature": {
8879         "sigtype": "oic.sec.sigtype.pk7",
8880         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8881     }
8882 }
8883 }
8884 ],
8885 "responses": {
8886     "400": {
8887         "description": "The request is invalid."
8888     },
8889     "201": {
8890         "description": "The ACL entry is created."
8891     },
8892     "204": {
8893         "description": "The ACL entry is updated."
8894     }
8895 }
8896 },
8897 "put": {
8898     "description": "Sets the sacl Resource data\n",
8899     "parameters": [
8900         { "$ref": "#/parameters/interface",
8901           {
8902             "name": "body",
8903             "in": "body",
8904             "required": true,
8905             "schema": { "$ref": "#/definitions/Sacl" },
8906             "x-example":
8907               {
8908                 "aclist2": [
8909                   {
8910                     "aceid": 1,
8911                     "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8912                     "resources": [
8913                       {
8914                         "href": "/temp",
8915                         "rt": ["oic.r.temperature"],
8916                         "if": ["oic.if.baseline", "oic.if.a"]
8917                       },
8918                       {
8919                         "href": "/temp",
8920                         "rt": ["oic.r.temperature"],
8921                         "if": ["oic.if.baseline", "oic.if.s"]
8922                       }
8923                     ],
8924                     "permission": 31,
8925                     "validity": [
8926                       {
8927                         "period": "20160101T180000Z/20170102T070000Z",
8928                         "recurrence": [ "DSTART:XXXXX",
8929 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8930                       },
8931                       {
8932                         "period": "20160101T180000Z/PT5H30M",
8933                         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8934                       }
8935                     ]
8936                   },
8937                   {
8938                     "aceid": 2,
8939                     "subject": {
8940                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8941                       "role": "SOME_STRING"
8942                     }

```

```

8943         "resources": [
8944             {
8945                 "href": "/light",
8946                 "rt": ["oic.r.light"],
8947                 "if": ["oic.if.baseline", "oic.if.a"]
8948             },
8949             {
8950                 "href": "/door",
8951                 "rt": ["oic.r.door"],
8952                 "if": ["oic.if.baseline", "oic.if.a"]
8953             }
8954         ],
8955         "permission": 15
8956     }
8957 ],
8958     "signature": {
8959         "sigtype": "oic.sec.sigtype.pk7",
8960         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8961     }
8962 }
8963 },
8964 ],
8965     "responses": {
8966         "400": {
8967             "description": "The request is invalid."
8968         },
8969         "201": {
8970             "description": "The signed ACL entry is created."
8971         }
8972     },
8973 },
8974     "delete": {
8975         "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
8976 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
8977 specified, only the matched entry is deleted.\n",
8978         "parameters": [
8979             { "$ref": "#/parameters/interface",
8980             {
8981                 "in": "query",
8982                 "description": "Delete the signed ACL identified by the string containing subject
8983 UUID.\n",
8984                 "type": "string",
8985                 "name": "subject"
8986             }
8987         ],
8988         "responses": {
8989             "200": {
8990                 "description": "The signed ACL instance or the the entire signed ACL Resource has
8991 been successfully deleted."
8992             },
8993             "400": {
8994                 "description": "The request is invalid."
8995             }
8996         }
8997     }
8998 },
8999 },
9000     "parameters": {
9001         "interface": {
9002             "in": "query",
9003             "name": "if",
9004             "type": "string",
9005             "enum": ["oic.if.baseline"]
9006         }
9007     },
9008     "definitions": {
9009         "Sacl": {
9010             "properties": {
9011                 "rt": {
9012                     "description": "Resource Type of the Resource.",
9013                     "items": {

```

```

9014         "maxLength": 64,
9015         "type": "string",
9016         "enum": ["oic.r.sacl"]
9017     },
9018     "minItems": 1,
9019     "readOnly": true,
9020     "type": "array"
9021 },
9022 "acllist2": {
9023     "description": "Access Control Entries in the ACL Resource.",
9024     "items": {
9025         "properties": {
9026             "aceid": {
9027                 "description": "An identifier for the ACE that is unique within the ACL. In cases
9028 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
9029                 "minimum": 1,
9030                 "type": "integer"
9031             },
9032             "permission": {
9033                 "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
9034 permissions.",
9035                 "x-detail-desc": [
9036                     "0 - No permissions.",
9037                     "1 - Create permission is granted.",
9038                     "2 - Read, observe, discover permission is granted.",
9039                     "4 - Write, update permission is granted.",
9040                     "8 - Delete permission is granted.",
9041                     "16 - Notify permission is granted."
9042                 ],
9043                 "maximum": 31,
9044                 "minimum": 0,
9045                 "type": "integer"
9046             },
9047             "resources": {
9048                 "description": "References the application's Resources to which a security policy
9049 applies.",
9050                 "items": {
9051                     "description": "Each Resource must have at least one of these properties set.",
9052                     "properties": {
9053                         "href": {
9054                             "allOf": [
9055                                 {
9056                                     "description": "When present, the ACE only applies when the href matches."
9057                                 },
9058                                 {
9059                                     "description": "This is the target URI, it can be specified as a Relative
9060 Reference or fully-qualified URI.",
9061                                     "format": "uri",
9062                                     "maxLength": 256,
9063                                     "type": "string"
9064                                 }
9065                             ]
9066                         },
9067                         "if": {
9068                 "description": "When present, the ACE only applies when the if (interface)
9069 matches\nThe interface set supported by this Resource.",
9070                 "items": {
9071                     "enum": [
9072                         "oic.if.baseline",
9073                         "oic.if.ll",
9074                         "oic.if.b",
9075                         "oic.if.rw",
9076                         "oic.if.r",
9077                         "oic.if.a",
9078                         "oic.if.s"
9079                     ],
9080                     "type": "string"
9081                 },
9082                 "minItems": 1,
9083                 "type": "array"
9084             },

```



```

9085         "rt": {
9086             "description": "When present, the ACE only applies when the rt (resource type)
9087 matches\nResource Type of the Resource.",
9088             "items": {
9089                 "maxLength": 64,
9090                 "type": "string"
9091             },
9092             "minItems": 1,
9093             "type": "array"
9094         },
9095         "wc": {
9096             "description": "A wildcard matching policy.",
9097             "pattern": "^[+*]$",
9098             "type": "string"
9099         }
9100     },
9101     "type": "object"
9102 },
9103 "type": "array"
9104 },
9105 "subject": {
9106     "anyOf": [
9107         {
9108             "description": "Device identifier.",
9109             "properties": {
9110                 "uuid": {
9111                     "description": "A UUID Device ID\nFormat pattern according to IETF RFC
9112 4122.",
9113                     "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
9114 fA-F0-9]{12}$",
9115                     "type": "string"
9116                 }
9117             },
9118             "required": [
9119                 "uuid"
9120             ],
9121             "type": "object"
9122         },
9123         {
9124             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
9125 <Authority> refers to the local entity or Device.",
9126             "properties": {
9127                 "authority": {
9128                     "description": "The Authority component of the entity being identified. A
9129 NULL <Authority> refers to the local entity or Device.",
9130                     "type": "string"
9131                 },
9132                 "role": {
9133                     "description": "The ID of the role being identified.",
9134                     "type": "string"
9135                 }
9136             },
9137             "required": [
9138                 "role"
9139             ],
9140             "type": "object"
9141         },
9142         {
9143             "properties": {
9144                 "conntype": {
9145                     "description": "This property allows an ACE to be matched based on the
9146 connection or message type.",
9147                     "x-detail-desc": [
9148                         "auth-crypt - ACE applies if the Client is authenticated and the data
9149 channel or message is encrypted and integrity protected.",
9150                         "anon-clear - ACE applies if the Client is not authenticated and the data
9151 channel or message is not encrypted but may be integrity protected."
9152                     ],
9153                     "enum": [
9154                         "auth-crypt",
9155                         "anon-clear"

```

```

9156         ],
9157         "type": "string"
9158     },
9159 },
9160     "required": [
9161         "conntype"
9162     ],
9163     "type": "object"
9164 }
9165 ]
9166 },
9167 "validity": {
9168     "description": "validity is an array of time-pattern objects.",
9169     "items": {
9170         "description": "The time-pattern contains a period and recurrence expressed in
9171 RFC5545 syntax.",
9172         "properties": {
9173             "period": {
9174                 "description": "String represents a period using the RFC5545 Period.",
9175                 "type": "string"
9176             },
9177             "recurrence": {
9178                 "description": "String array represents a recurrence rule using the RFC5545
9179 Recurrence.",
9180                 "items": {
9181                     "type": "string"
9182                 },
9183                 "type": "array"
9184             }
9185         },
9186         "required": [
9187             "period"
9188         ],
9189         "type": "object"
9190     },
9191     "type": "array"
9192 }
9193 },
9194 "required": [
9195     "aceid",
9196     "resources",
9197     "permission",
9198     "subject"
9199 ],
9200 "type": "object"
9201 },
9202 "type": "array"
9203 },
9204 "n": {
9205     "$ref":
9206 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9207 schema.json#/definitions/n"
9208 },
9209 "id": {
9210     "$ref":
9211 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9212 schema.json#/definitions/id"
9213 },
9214 "signature": {
9215     "description": "The signature over the ACL Resource\nEncoded signature data.",
9216     "properties": {
9217         "sigtype": {
9218             "description": "The string specifies the predefined signature format.",
9219             "x-detail-desc": [
9220                 "RFC7515 JSON web signature (JWS) object.",
9221                 "RFC2315 base64 encoded object.",
9222                 "CBOR encoded JWS object."
9223             ],
9224             "enum": [
9225                 "oic.sec.sigtype.jws",
9226                 "oic.sec.sigtype.pk7",

```

```

9227         "oic.sec.sigtype.cws"
9228     ],
9229     "type": "string"
9230 },
9231 "sigvalue": {
9232     "description": "The encoded signature.",
9233     "type": "string"
9234 }
9235 },
9236 "required": [
9237     "sigtype",
9238     "sigvalue"
9239 ],
9240 "type": "object"
9241 },
9242 "if": {
9243     "description": "The interface set supported by this Resource.",
9244     "items": {
9245         "enum": [
9246             "oic.if.baseline"
9247         ],
9248         "type": "string"
9249     },
9250     "minItems": 1,
9251     "readOnly": true,
9252     "type": "array"
9253 },
9254 },
9255 "type" : "object",
9256 "required": ["aclist2", "signature"]
9257 }
9258 }
9259 }
9260

```

### 9261 C.12.5 Property definition

9262 Table C.22 defines the Properties that are part of the "oic.r.sacl" Resource Type.

9263 **Table C.22 – The Property definitions of the Resource with type "rt" = "oic.r.sacl".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
id	multiple types: see schema	No	Read Write	
signature	object: see schema	Yes	Read Write	The signature over the ACL Resource Encoded signature data.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the ACL Resource.
n	multiple types: see schema	No	Read Write	

### 9264 C.12.6 CRUDN behaviour

9265 Table C.23 defines the CRUDN operations that are supported on the "oic.r.sacl" Resource Type.

9266 **Table C.23 – The CRUDN operations of the Resource with type "rt" = "oic.r.sacl".**

Create	Read	Update	Delete	Notify
put	get	post	delete	observe

9267 **C.13 Session**

9268 **C.13.1 Introduction**

9269 Resource that manages the persistent session between a Device and OCF Cloud.

9270 **C.13.2 Well-known URI**

9271 /oic/sec/session

9272 **C.13.3 Resource type**

9273 The Resource Type is defined as: "oic.r.session".

9274 **C.13.4 OpenAPI 2.0 definition**

```

9275 {
9276   "swagger": "2.0",
9277   "info": {
9278     "title": "Session",
9279     "version": "v1.0-20181001",
9280     "license": {
9281       "name": "OCF Data Model License",
9282       "url":
9283         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9284         CENSE.md",
9285       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9286         reserved."
9287     },
9288     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9289   },
9290   "schemes": ["http"],
9291   "consumes": ["application/json"],
9292   "produces": ["application/json"],
9293   "paths": {
9294     "/oic/sec/session" : {
9295       "post": {
9296         "description": "Resource that manages the persistent session between a Device and OCF
9297         Cloud.",
9298         "parameters": [
9299           { "$ref": "#/parameters/interface" },
9300           {
9301             "name": "body",
9302             "in": "body",
9303             "required": true,
9304             "schema": { "$ref": "#/definitions/Account-Session-Request" },
9305             "x-example":
9306               {
9307                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9308                 "di" : "9cfbeb8e-5ale-4dlc-9d01-00c04fd430c8",
9309                 "accesstoken" : "0f3d9f7fe5491d54077d",
9310                 "login" : true
9311               }
9312           }
9313         ],
9314         "responses": {
9315           "204": {
9316             "description": "",
9317             "x-example":
9318               {
9319                 "rt": ["oic.r.session"],
9320                 "expiresin" : 3600
9321               },
9322             "schema": { "$ref": "#/definitions/Account-Session-Response" }
9323           }
9324         }
9325       }
9326     }
9327   }

```

```

9325     }
9326   },
9327 },
9328 "parameters": {
9329   "interface" : {
9330     "in" : "query",
9331     "name" : "if",
9332     "type" : "string",
9333     "enum" : ["oic.if.baseline"]
9334   }
9335 },
9336 "definitions": {
9337   "Account-Session-Request" : {
9338     "properties": {
9339       "uid": {
9340         "description": "Format pattern according to IETF RFC 4122.",
9341         "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}$",
9342         "type": "string"
9343       },
9344       "di": {
9345         "description": "The Device ID\nFormat pattern according to IETF RFC 4122.",
9346         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9347         "type": "string"
9348       },
9349       "accesstoken": {
9350         "description": "Access-Token used to grant access right for the Device to sign-in.",
9351         "pattern": "(?!$|\\s+).*",
9352         "type": "string"
9353       },
9354       "login": {
9355         "description": "Action for the request: true = login, false = logout.",
9356         "type": "boolean"
9357     }
9358   },
9359   "type" : "object",
9360   "required": ["uid", "di", "accesstoken", "login"]
9361 },
9362 "Account-Session-Response" : {
9363   "properties": {
9364     "expiresin": {
9365       "description": "Access-Token remaining life time in seconds (-1 if permanent).",
9366       "readOnly": true,
9367       "type": "integer"
9368     },
9369     "rt": {
9370       "description": "Resource Type of the Resource.",
9371       "items": {
9372         "maxLength": 64,
9373         "type": "string",
9374         "enum": ["oic.r.session"]
9375       },
9376       "minItems": 1,
9377       "readOnly": true,
9378       "type": "array"
9379     },
9380     "n": {
9381       "$ref":
9382 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9383 schema.json#/definitions/n"
9384     },
9385     "id": {
9386       "$ref":
9387 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9388 schema.json#/definitions/id"
9389     },
9390     "if": {
9391       "description": "The interface set supported by this Resource.",
9392       "items": {
9393         "enum": [
9394           "oic.if.baseline"
9395         ],

```

```

9396     "type": "string"
9397   },
9398   "minItems": 1,
9399   "readOnly": true,
9400   "type": "array"
9401 }
9402 },
9403 "type" : "object",
9404 "required" : ["expiresin"]
9405 }
9406 }
9407 }
9408

```

9409 **C.13.5 Property definition**

9410 Table C.24 defines the Properties that are part of the "oic.r.session" Resource Type.

9411 **Table C.24 – The Property definitions of the Resource with type "rt" = "oic.r.session".**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The interface set supported by this Resource.
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.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
di	string	Yes	Read Write	The Device ID Format pattern according to IETF RFC 4122.
accesstoken	string	Yes	Read Write	Access-Token used to grant access right for the Device to sign-in.
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.

9412 **C.13.6 CRUDN behaviour**

9413 Table C.25 defines the CRUDN operations that are supported on the "oic.r.session" Resource Type.

9415 **Table C.25 – The CRUDN operations of the Resource with type "rt" = "oic.r.session".**

Create	Read	Update	Delete	Notify
		post		

## 9416 C.14 Security Profile

### 9417 C.14.1 Introduction

9418 Resource specifying supported and active security profile(s).

9419

### 9420 C.14.2 Well-known URI

9421 /oic/sec/sp

### 9422 C.14.3 Resource type

9423 The Resource Type is defined as: "oic.r.sp".

### 9424 C.14.4 OpenAPI 2.0 definition

```
9425 {
9426   "swagger": "2.0",
9427   "info": {
9428     "title": "Security Profile",
9429     "version": "v1.0-20190208",
9430     "license": {
9431       "name": "OCF Data Model License",
9432       "url":
9433         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9434         CENSE.md",
9435       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9436         reserved."
9437     },
9438     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9439   },
9440   "schemes": ["http"],
9441   "consumes": ["application/json"],
9442   "produces": ["application/json"],
9443   "paths": {
9444     "/oic/sec/sp" : {
9445       "get": {
9446         "description": "Resource specifying supported and active security profile(s).\n",
9447         "parameters": [
9448           {"$ref": "#/parameters/interface"}
9449         ],
9450         "responses": {
9451           "200": {
9452             "description": "",
9453             "x-example":
9454               {
9455                 "rt": ["oic.r.sp"],
9456                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9457                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9458               },
9459             "schema": { "$ref": "#/definitions/SP" }
9460           },
9461           "400": {
9462             "description": "The request is invalid."
9463           }
9464         }
9465       },
9466       "post": {
9467         "description": "Sets or updates Device provisioning status data.\n",
9468         "parameters": [
9469           {"$ref": "#/parameters/interface"},
9470           {
9471             "name": "body",
9472             "in": "body",
9473             "required": true,
9474             "schema": { "$ref": "#/definitions/SP-Update" },
9475             "x-example":
9476               {
9477                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9478                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9479               }
9480           }
9481         ]
9482       }
9483     }
9484   }
9485 }
```

```

9479     }
9480   }
9481 ],
9482   "responses": {
9483     "200": {
9484       "description": "",
9485       "x-example":
9486         {
9487           "rt": ["oic.r.sp"],
9488           "supportedprofiles": ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9489           "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9490         },
9491     "schema": { "$ref": "#/definitions/SP" }
9492   },
9493   "400": {
9494     "description": "The request is invalid."
9495   }
9496 }
9497 }
9498 }
9499 },
9500 "parameters": {
9501   "interface" : {
9502     "in" : "query",
9503     "name" : "if",
9504     "type" : "string",
9505     "enum" : ["oic.if.baseline"]
9506   }
9507 },
9508 "definitions": {
9509   "SP" : {
9510     "properties": {
9511       "rt": {
9512         "description": "Resource Type of the Resource.",
9513         "items": {
9514           "maxLength": 64,
9515           "type": "string",
9516           "enum": ["oic.r.sp"]
9517         },
9518         "minItems": 1,
9519         "readOnly": true,
9520         "type": "array"
9521       },
9522       "n": {
9523         "$ref":
9524 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9525 schema.json#/definitions/n"
9526       },
9527       "id": {
9528         "$ref":
9529 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9530 schema.json#/definitions/id"
9531       },
9532       "currentprofile": {
9533         "description": "Security Profile currently active.",
9534         "type": "string"
9535       },
9536       "supportedprofiles": {
9537         "description": "Array of supported Security Profiles.",
9538         "items": {
9539           "type": "string"
9540         },
9541         "type": "array"
9542       },
9543       "if": {
9544         "description": "The interface set supported by this Resource.",
9545         "items": {
9546           "enum": [
9547             "oic.if.baseline"
9548           ],
9549         "type": "string"

```



```

9550     },
9551     "minItems": 1,
9552     "readOnly": true,
9553     "type": "array"
9554   }
9555 },
9556 "type" : "object",
9557 "required": ["supportedprofiles", "currentprofile"]
9558 },
9559 "SP-Update" : {
9560   "properties": {
9561     "currentprofile": {
9562       "description": "Security Profile currently active.",
9563       "type": "string"
9564     },
9565     "supportedprofiles": {
9566       "description": "Array of supported Security Profiles.",
9567       "items": {
9568         "type": "string"
9569       },
9570       "type": "array"
9571     }
9572   },
9573   "type" : "object"
9574 }
9575 }
9576 }
9577

```

### C.14.5 Property definition

Table C.26 defines the Properties that are part of the "oic.r.sp" Resource Type.

**Table C.26 – 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	
n	multiple types: see schema	No	Read Write	
currentprofile	string	Yes	Read Write	Security Profile currently active.
supportedprofiles	array: see schema	Yes	Read Write	Array of supported Security Profiles.
rt	array: see schema	No	Read Only	Resource Type of the Resource.
if	array: see schema	No	Read Only	The interface set supported by this Resource.

### C.14.6 CRUDN behaviour

Table C.27 defines the CRUDN operations that are supported on the "oic.r.sp" Resource Type.

**Table C.27 – The CRUDN operations of the Resource with type "rt" = "oic.r.sp".**

Create	Read	Update	Delete	Notify
	get	post		observe

## 9584 C.15 Token Refresh

### 9585 C.15.1 Introduction

9586 Obtain fresh access-token using the refresh token, client should refresh access-token before it  
9587 expires.

### 9588 C.15.2 Well-known URI

9589 /oic/sec/tokenrefresh

### 9590 C.15.3 Resource type

9591 The Resource Type is defined as: "oic.r.tokenrefresh".

### 9592 C.15.4 OpenAPI 2.0 definition

```
9593 {
9594   "swagger": "2.0",
9595   "info": {
9596     "title": "Token Refresh",
9597     "version": "v1.0-20181001",
9598     "license": {
9599       "name": "OCF Data Model License",
9600       "url":
9601         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9602         CENSE.md",
9603       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9604         reserved."
9605     },
9606     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9607   },
9608   "schemes": ["http"],
9609   "consumes": ["application/json"],
9610   "produces": ["application/json"],
9611   "paths": {
9612     "/oic/sec/tokenrefresh" : {
9613       "post": {
9614         "description": "Obtain fresh access-token using the refresh token, client should refresh
9615         access-token before it expires.\n",
9616         "parameters": [
9617           { "$ref": "#/parameters/interface" },
9618           {
9619             "name": "body",
9620             "in": "body",
9621             "required": true,
9622             "schema": { "$ref": "#/definitions/TokenRefresh-Request" },
9623             "x-example":
9624               {
9625                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9626                 "di" : "9cfbeb8e-5ale-4d1c-9d01-00c04fd430c8",
9627                 "refreshToken" : "00fe4644a6fbe5324eec"
9628               }
9629           }
9630         ],
9631         "responses": {
9632           "204": {
9633             "description" : "2.04 Changed respond with new access-token.\n",
9634             "x-example":
9635               {
9636                 "rt": ["oic.r.tokenrefresh"],
9637                 "accesstoken" : "8ce598980761869837be",
9638                 "refreshToken" : "d4922312b6df0518e146",
9639                 "expiresin" : 3600
9640               }
9641           },
9642           "schema": { "$ref": "#/definitions/TokenRefresh-Response" }
9643         }
9644       }
9645     }
9646   }
```

```

9647 },
9648 "parameters": {
9649   "interface" : {
9650     "in" : "query",
9651     "name" : "if",
9652     "type" : "string",
9653     "enum" : ["oic.if.baseline"]
9654   }
9655 },
9656 "definitions": {
9657   "TokenRefresh-Request" : {
9658     "properties": {
9659       "refreshtoken": {
9660         "description": "Refresh token received by account management or during token refresh
9661 procedure.",
9662         "pattern": "(?!$|\\s+).*",
9663         "type": "string"
9664       },
9665       "uid": {
9666         "description": "Format pattern according to IETF RFC 4122.",
9667         "pattern": "^-[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9668         "type": "string"
9669       },
9670       "di": {
9671         "description": "Format pattern according to IETF RFC 4122.",
9672         "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}$",
9673         "type": "string"
9674       }
9675     },
9676     "type" : "object",
9677     "required": ["uid", "di", "refreshtoken"]
9678   },
9679   "TokenRefresh-Response" : {
9680     "properties": {
9681       "expiresin": {
9682         "description": "Access-Token life time in seconds (-1 if permanent).",
9683         "readOnly": true,
9684         "type": "integer"
9685       },
9686       "rt": {
9687         "description": "Resource Type of the Resource.",
9688         "items": {
9689           "maxLength": 64,
9690           "type": "string",
9691           "enum": ["oic.r.tokenrefresh"]
9692         },
9693         "minItems": 1,
9694         "readOnly": true,
9695         "type": "array"
9696       },
9697       "refreshtoken": {
9698         "description": "Refresh token received by account management or during token refresh
9699 procedure.",
9700         "pattern": "(?!$|\\s+).*",
9701         "type": "string"
9702       },
9703       "accesstoken": {
9704         "description": "Granted Access-Token.",
9705         "pattern": "(?!$|\\s+).*",
9706         "readOnly": true,
9707         "type": "string"
9708       },
9709       "n": {
9710         "$ref":
9711 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9712 schema.json#/definitions/n"
9713       },
9714       "id": {
9715         "$ref":
9716 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9717 schema.json#/definitions/id"

```

```

9718     },
9719     "if" :
9720     {
9721         "description": "The interface set supported by this Resource.",
9722         "items": {
9723             "enum": [
9724                 "oic.if.baseline"
9725             ],
9726             "type": "string"
9727         },
9728         "minItems": 1,
9729         "readOnly": true,
9730         "type": "array"
9731     }
9732 },
9733 "type" : "object",
9734 "required": ["accesstoken", "refreshtoken", "expiresin"]
9735 }
9736 }
9737 }
9738

```

### 9739 C.15.5 Property definition

9740 Table C.28 defines the Properties that are part of the "oic.r.tokenrefresh" Resource Type.

9741 **Table C.28 – The Property definitions of the Resource with type "rt" = "oic.r.tokenrefresh".**

Property name	Value type	Mandatory	Access mode	Description
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.
if	array: see schema	No	Read Only	The interface set supported by this Resource.
expiresin	integer	Yes	Read Only	Access-Token life time in seconds (-1 if permanent).
accesstoken	string	Yes	Read Only	Granted Access-Token.
refreshtoken	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure.
n	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource.

id	multiple types: see schema	No	Read Write	
----	-------------------------------	----	------------	--

9742 **C.15.6 CRUDN behaviour**

9743 Table C.29 defines the CRUDN operations that are supported on the "oic.r.tokenrefresh"  
9744 Resource Type.

9745 **Table C.29 – The CRUDN operations of the Resource with type "rt" = "oic.r.tokenrefresh".**

Create	Read	Update	Delete	Notify
		post		

9746 **Annex D**  
9747 **(informative)**

9748 **OID definitions**  
9749

9750 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to  
9751 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of  
9752 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
9753 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
9754     private(4) enterprise(1) OCF(51414) }
9755
9756 -- OCF Security specific OIDs
9757
9758 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
9759 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9760
9761 -- OCF Security Categories
9762
9763 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
9764 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
9765
9766 -- OCF Security Profiles
9767
9768 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
9769 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
9770 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
9771 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
9772 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
9773
9774 sp-unspecified-v0 ::= ocfSecurityProfileOID {id-sp-unspecified 0}
9775 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
9776 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
9777 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
9778 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
9779
9780 ocfSecurityProfileOID ::= UTF8String
9781
9782 -- OCF Security Certificate Policies
9783
9784 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
9785
9786 -- OCF X.509v3 Extensions
9787
9788 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9789 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
9790 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
9791 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
9792
9793 ocfVersion ::= SEQUENCE {
9794     major    INTEGER,
9795     minor    INTEGER,
9796     build    INTEGER}
9797
9798 ocfCompliance ::= SEQUENCE {
9799     version      ocfVersion,
9800     securityProfileSEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
9801     deviceName   UTF8String,
9802     deviceManufacturer UTF8String}
9803
9804 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
9805 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
9806
9807 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
9808
9809 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
9810
9811 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
9812 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
9813 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
9814
9815 ocfCPLAttributes ::= SEQUENCE {
9816     cpl-at-IANAPen UTF8String,
9817     cpl-at-model UTF8String,
9818     cpl-at-version UTF8String}
```

**Annex E  
(informative)**

**Security considerations specific to Bridged Protocols**

9819  
9820  
9821  
9822

9823 The text in this Annex is provided for information only. This Annex has no normative impact. This  
9824 information is applicable at the time of initial publication and may become out of date.

**E.1 Security Considerations specific to the AllJoyn Protocol**

9825 This clause intentionally left empty.

**E.2 Security Considerations specific to the Bluetooth LE Protocol**

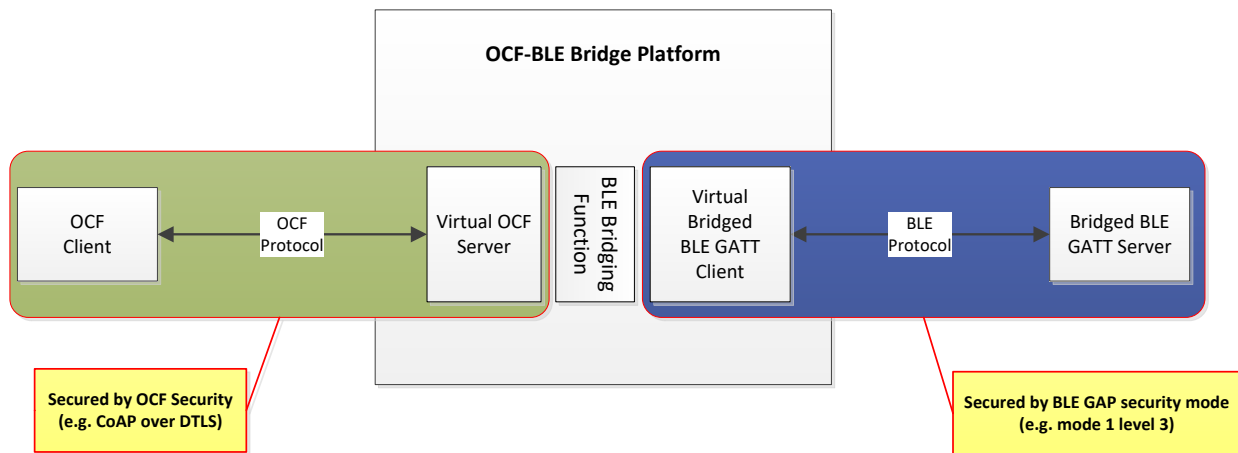
9828 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security  
9829 mode has several security levels (see Table E.1)

9830 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF  
9831 perspective. The appropriate selection of security mode and level is left to the vendor.

9832 **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)

9833 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge Platform are  
9834 secured by their own security.



9835  
9836

**Figure E-1 Security Considerations for BLE Bridge**

**E.3 Security Considerations specific to the oneM2M Protocol**

9839 This clause intentionally left empty.



9840 **E.4 Security Considerations specific to the U+ Protocol**

9841 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

9842 **Table E.2 TLS 1.2 Cipher Suites used by U+**

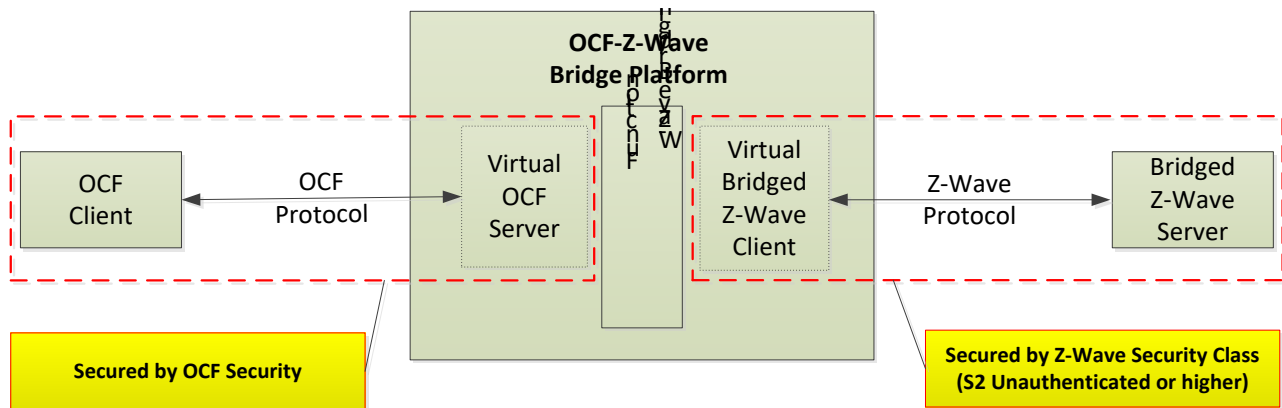
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

9843 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

9844 **E.5 Security Considerations specific to the Z-Wave Protocol**

9845 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2  
 9846 Security Class, as shown in Table E.3. Bridged Z-wave Servers using S2 Security Class for  
 9847 communication with a Virtual Bridged Client would typically be considered secure from an OCF  
 9848 perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

9849 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their  
 9850 own security.



9851  
9852

9853 **Figure E-2 Security Considerations for Z-Wave Bridge**

9854 All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2  
9855 Unauthenticated provides the following advantages from the security perspective;

- 9856 – The unique device specific key for every secure device enables validation of device identity  
9857 and prevents man-in-the-middle compromises to security
- 9858 – The Secure cryptographic key exchange methods during inclusion achieves high level of  
9859 security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.
- 9860 – Out of band key exchange for product authentication which is combined with device specific  
9861 key prevents eavesdropping and man-in-the-middle attack vectors.

9862 See Table E.3 for a summary of Z-Wave Security Classes.

9863 **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

9864 On the other hand, S0 Security Class has the vulnerability of security during inclusion by  
9865 exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the  
9866 disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices  
9867 might be no longer secure in that case.

9868 **E.6 Security Considerations specific to the Zigbee Protocol**

9869 The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the  
9870 network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0  
9871 stack, "nwkSecurityLevel", represents the security level of a device.

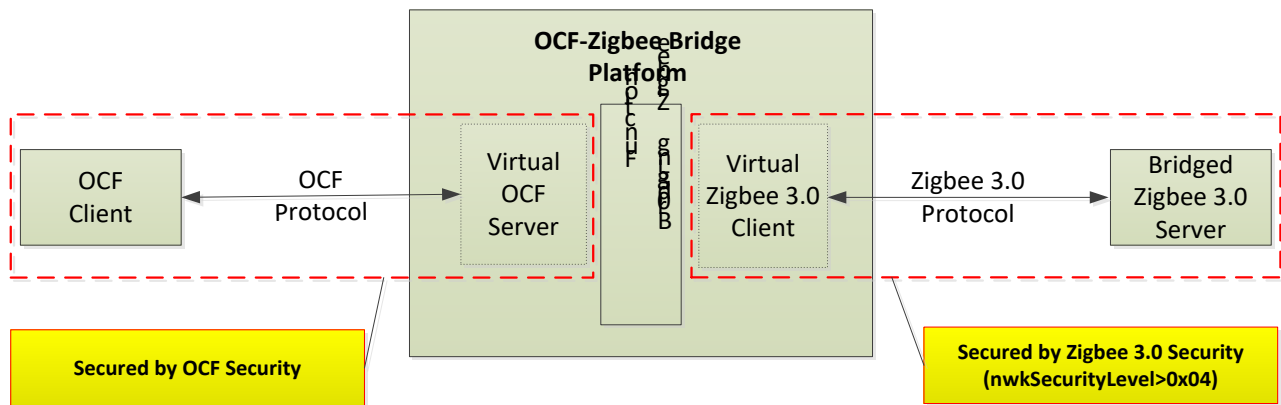
9872 The security level `nwkSecurityLevel > 0x04` provides message integrity code (MIC) and/or  
 9873 AES128-CCM encryption (ENC). Zigbee Servers using `nwkSecurityLevel > 0x04` would typically  
 9874 be considered secure from an OCF perspective. The appropriate selection for `nwkSecurityLevel`  
 9875 is left to the vendor.

9876 See Table E.4 for a summary of the Zigbee Security Levels.

9877 **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)

9878 Figure E-3 shows how communications in both ecosystems of OCF-Zigbee Bridge Platform are  
 9879 secured by their own security.



9880  
 9881

9882 **Figure E-3 Security Considerations for Zigbee Bridge**