

OCF Security Specification

VERSION 2.0.1 | February 11, 2019



CONTACT admin@openconnectivity.org
Copyright OCF © 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 © 2017-2019 Open Connectivity Foundation, Inc. All rights reserved.

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

17	1	Scope.....	1
18	2	Normative References	1
19	3	Terms, definitions, and abbreviated terms	3
20	3.1	Terms and definitions.....	3
21	3.2	Abbreviated terms.....	6
22	4	Document Conventions and Organization	9
23	4.1	Conventions.....	9
24	4.2	Notation.....	10
25	4.3	Data types	11
26	4.4	Document structure.....	11
27	5	Security Overview.....	12
28	5.1	Preamble	12
29	5.2	Access Control.....	14
30	5.2.1	ACL Architecture	16
31	5.2.2	Access Control Scoping Levels.....	19
32	5.3	Onboarding Overview	21
33	5.3.1	Onboarding General	21
34	5.3.2	OnBoarding Steps	23
35	5.3.3	Establishing a Device Owner	24
36	5.3.4	Provisioning for Normal Operation	25
37	5.3.5	Device Provisioning for OCF Cloud and Device Registration Overview	25
38	5.3.6	OCF Compliance Management System.....	25
39	5.4	Provisioning.....	26
40	5.4.1	Provisioning General	26
41	5.4.2	Provisioning other services.....	26
42	5.4.3	Provisioning Credentials for Normal Operation	27
43	5.4.4	Role Assignment and Provisioning for Normal Operation	27
44	5.4.5	ACL provisioning	27
45	5.5	Secure Resource Manager (SRM).....	27
46	5.6	Credential Overview.....	28
47	6	Security for the Discovery Process	29
48	6.1	Preamble	29
49	6.2	Security Considerations for Discovery.....	29
50	7	Security Provisioning.....	32
51	7.1	Device Identity.....	32
52	7.1.1	General Device Identity	32
53	7.1.2	Device Identity for Devices with UAID.....	32
54	7.2	Device Ownership.....	34
55	7.3	Device Ownership Transfer Methods.....	34
56	7.3.1	OTM implementation requirements	34
57	7.3.2	SharedKey Credential Calculation	36
58	7.3.3	Certificate Credential Generation.....	36
59	7.3.4	Just-Works OTM.....	36

60	7.3.5	Random PIN Based OTM.....	38
61	7.3.6	Manufacturer Certificate Based OTM.....	41
62	7.3.7	Vendor Specific OTMs.....	43
63	7.3.8	Establishing Owner Credentials.....	44
64	7.3.9	Security considerations regarding selecting an Ownership Transfer Method ..	52
65	7.3.10	Security Profile Assignment.....	52
66	7.4	Provisioning.....	53
67	7.4.1	Provisioning Flows.....	53
68	7.5	Device Provisioning for OCF Cloud.....	58
69	7.5.1	Cloud Provisioning General.....	58
70	7.5.2	Device Provisioning by Mediator.....	58
71	8	Device Onboarding State Definitions.....	59
72	8.1	Device Onboarding General.....	59
73	8.2	Device Onboarding-Reset State Definition.....	61
74	8.3	Device Ready-for-OTM State Definition.....	61
75	8.4	Device Ready-for-Provisioning State Definition.....	62
76	8.5	Device Ready-for-Normal-Operation State Definition.....	62
77	8.6	Device Soft Reset State Definition.....	63
78	9	Security Credential Management.....	65
79	9.1	Preamble.....	65
80	9.2	Credential Lifecycle.....	65
81	9.2.1	Credential Lifecycle General.....	65
82	9.2.2	Creation.....	65
83	9.2.3	Deletion.....	65
84	9.2.4	Refresh.....	65
85	9.2.5	Revocation.....	65
86	9.3	Credential Types.....	66
87	9.3.1	Preamble.....	66
88	9.3.2	Pair-wise Symmetric Key Credentials.....	66
89	9.3.3	Group Symmetric Key Credentials.....	66
90	9.3.4	Asymmetric Authentication Key Credentials.....	67
91	9.3.5	Asymmetric Key Encryption Key Credentials.....	67
92	9.3.6	Certificate Credentials.....	68
93	9.3.7	Password Credentials.....	68
94	9.4	Certificate Based Key Management.....	68
95	9.4.1	Overview.....	68
96	9.4.2	X.509 Digital Certificate Profiles.....	69
97	9.4.3	Certificate Revocation List (CRL) Profile.....	78
98	9.4.4	Resource Model.....	79
99	9.4.5	Certificate Provisioning.....	79
100	9.4.6	CRL Provisioning.....	80
101	10	Device Authentication.....	83
102	10.1	Device Authentication General.....	83
103	10.2	Device Authentication with Symmetric Key Credentials.....	83

104	10.3	Device Authentication with Raw Asymmetric Key Credentials.....	83
105	10.4	Device Authentication with Certificates	83
106	10.4.1	Device Authentication with Certificates General.....	83
107	10.4.2	Role Assertion with Certificates	84
108	10.4.3	OCF PKI Roots.....	85
109	10.4.4	PKI Trust Store.....	85
110	10.4.5	Path Validation and extension processing.....	86
111	10.5	Device Authentication with OCF Cloud.....	87
112	10.5.1	Device Authentication with OCF Cloud General	87
113	10.5.2	Device Connection with the OCF Cloud	87
114	10.5.3	Security Considerations.....	88
115	11	Message Integrity and Confidentiality	90
116	11.1	Preamble	90
117	11.2	Session Protection with DTLS.....	90
118	11.2.1	DTLS Protection General.....	90
119	11.2.2	Unicast Session Semantics.....	90
120	11.2.3	Cloud Session Semantics	90
121	11.3	Cipher Suites	90
122	11.3.1	Cipher Suites General	90
123	11.3.2	Cipher Suites for Device Ownership Transfer	90
124	11.3.3	Cipher Suites for Symmetric Keys.....	91
125	11.3.4	Cipher Suites for Asymmetric Credentials.....	92
126	11.3.5	Cipher suites for OCF Cloud Credentials	92
127	12	Access Control	94
128	12.1	ACL Generation and Management	94
129	12.2	ACL Evaluation and Enforcement.....	94
130	12.2.1	ACL Evaluation and Enforcement General	94
131	12.2.2	Host Reference Matching	94
132	12.2.3	Resource Wildcard Matching	94
133	12.2.4	Multiple Criteria Matching.....	95
134	12.2.5	Subject Matching using Wildcards	95
135	12.2.6	Subject Matching using Roles.....	95
136	12.2.7	ACL Evaluation.....	96
137	13	Security Resources	98
138	13.1	Security Resources General	98
139	13.2	Device Owner Transfer Resource	100
140	13.2.1	Device Owner Transfer Resource General.....	100
141	13.2.2	Persistent and Semi-persistent Device Identifiers	103
142	13.2.3	Onboarding Considerations for Device Identifier	103
143	13.2.4	OCF defined OTMs.....	104
144	13.3	Credential Resource	105
145	13.3.1	Credential Resource General.....	105
146	13.3.2	Properties of the Credential Resource	109
147	13.3.3	Key Formatting	112

148	13.3.4	Credential Refresh Method Details	112
149	13.4	Certificate Revocation List	114
150	13.4.1	CRL Resource Definition	114
151	13.5	ACL Resources.....	114
152	13.5.1	ACL Resources General	114
153	13.5.2	OCF Access Control List (ACL) BNF defines ACL structures.	114
154	13.5.3	ACL Resource	115
155	13.6	Access Manager ACL Resource.....	122
156	13.7	Signed ACL Resource.....	123
157	13.8	Provisioning Status Resource	124
158	13.9	Certificate Signing Request Resource.....	129
159	13.10	Roles Resource	130
160	13.11	Account Resource.....	131
161	13.12	Account Session resource.....	133
162	13.13	Account Token Refresh Resource	134
163	13.14	Security Virtual Resources (SVRs) and Access Policy	135
164	13.15	SVRs, Discoverability and OCF Endpoints	135
165	13.16	Additional Privacy Consideration for Core and SVRs Resources	135
166	13.16.1	Additional Privacy Considerations for Core and SVR Resources General ...	135
167	13.16.2	Privacy Protecting the Device Identifiers.....	138
168	13.16.3	Privacy Protecting the Protocol Independent Device Identifier	138
169	13.16.4	Privacy Protecting the Platform Identifier	138
170	13.17	Easy Setup Resource Device State.....	139
171	14	Security Hardening Guidelines/ Execution Environment Security	142
172	14.1	Preamble	142
173	14.2	Execution Environment Elements.....	142
174	14.2.1	Execution Environment Elements General	142
175	14.2.2	Secure Storage.....	142
176	14.2.3	Secure execution engine	145
177	14.2.4	Trusted input/output paths	145
178	14.2.5	Secure clock.....	145
179	14.2.6	Approved algorithms.....	145
180	14.2.7	Hardware tamper protection.....	146
181	14.3	Secure Boot.....	146
182	14.3.1	Concept of software module authentication.....	146
183	14.3.2	Secure Boot process	148
184	14.3.3	Robustness Requirements.....	148
185	14.4	Attestation	148
186	14.5	Software Update	148
187	14.5.1	Overview:	148
188	14.5.2	Recognition of Current Differences	149
189	14.5.3	Software Version Validation.....	149
190	14.5.4	Software Update.....	149
191	14.5.5	Recommended Usage.....	149

192	14.6	Non-OCF Endpoint interoperability.....	150
193	14.7	Security Levels	150
194	14.8	Security Profiles.....	151
195	14.8.1	Security Profiles General.....	151
196	14.8.2	Identification of Security Profiles (Normative)	151
197	14.8.3	Security Profiles	153
198	15	Device Type Specific Requirements.....	158
199	15.1	Bridging Security	158
200	15.1.1	Universal Requirements for Bridging to another Ecosystem	158
201	15.1.2	Additional Security Requirements specific to Bridged Protocols	158
202		Annex A (informative) Access Control Examples.....	160
203	A.1	Example OCF ACL Resource	160
204	A.2	Example AMS	160
205		Annex B (Informative) Execution Environment Security Profiles	162
206		Annex C (normative) Resource Type definitions.....	163
207	C.1	List of Resource Type definitions	163
208	C.2	Account Token.....	163
209	C.2.1	Introduction	163
210	C.2.2	Well-known URI.....	163
211	C.2.3	Resource type	163
212	C.2.4	OpenAPI 2.0 definition.....	163
213	C.2.5	Property definition	166
214	C.2.6	CRUDN behaviour	167
215	C.3	Access Control List.....	167
216	C.3.1	Introduction	167
217	C.3.2	Well-known URI.....	167
218	C.3.3	Resource type	167
219	C.3.4	OpenAPI 2.0 definition.....	167
220	C.3.5	Property definition	174
221	C.3.6	CRUDN behaviour	174
222	C.4	Access Control List-2.....	175
223	C.4.1	Introduction	175
224	C.4.2	Well-known URI.....	175
225	C.4.3	Resource type	175
226	C.4.4	OpenAPI 2.0 definition.....	175
227	C.4.5	Property definition	184
228	C.4.6	CRUDN behaviour	185
229	C.5	Managed Access Control	185
230	C.5.1	Introduction	185
231	C.5.2	Well-known URI.....	185
232	C.5.3	Resource type	185
233	C.5.4	OpenAPI 2.0 definition.....	185
234	C.5.5	Property definition	188
235	C.5.6	CRUDN behaviour	189

236	C.6	Credential	189
237	C.6.1	Introduction	189
238	C.6.2	Well-known URI	189
239	C.6.3	Resource type	189
240	C.6.4	OpenAPI 2.0 definition.....	189
241	C.6.5	Property definition	199
242	C.6.6	CRUDN behaviour	200
243	C.7	Certificate Revocation.....	200
244	C.7.1	Introduction	200
245	C.7.2	Well-known URI	200
246	C.7.3	Resource type	200
247	C.7.4	OpenAPI 2.0 definition.....	200
248	C.7.5	Property definition	202
249	C.7.6	CRUDN behaviour	203
250	C.8	Certificate Signing Request.....	203
251	C.8.1	Introduction	203
252	C.8.2	Well-known URI	203
253	C.8.3	Resource type	203
254	C.8.4	OpenAPI 2.0 definition.....	203
255	C.8.5	Property definition	205
256	C.8.6	CRUDN behaviour	205
257	C.9	Device Owner Transfer Method.....	205
258	C.9.1	Introduction	205
259	C.9.2	Well-known URI	206
260	C.9.3	Resource type	206
261	C.9.4	OpenAPI 2.0 definition.....	206
262	C.9.5	Property definition	209
263	C.9.6	CRUDN behaviour	211
264	C.10	Device Provisioning Status	211
265	C.10.1	Introduction	211
266	C.10.2	Well-known URI	211
267	C.10.3	Resource type	211
268	C.10.4	OpenAPI 2.0 definition.....	211
269	C.10.5	Property definition	215
270	C.10.6	CRUDN behaviour	220
271	C.11	Asserted Roles	220
272	C.11.1	Introduction	220
273	C.11.2	Well-known URI	220
274	C.11.3	Resource type	220
275	C.11.4	OpenAPI 2.0 definition.....	220
276	C.11.5	Property definition	229
277	C.11.6	CRUDN behaviour	229
278	C.12	Signed Access Control List	229
279	C.12.1	Introduction	229

280	C.12.2	Well-known URI	230
281	C.12.3	Resource type	230
282	C.12.4	OpenAPI 2.0 definition.....	230
283	C.12.5	Property definition	237
284	C.12.6	CRUDN behaviour	238
285	C.13	Session.....	238
286	C.13.1	Introduction	238
287	C.13.2	Well-known URI.....	238
288	C.13.3	Resource type	238
289	C.13.4	OpenAPI 2.0 definition.....	238
290	C.13.5	Property definition	240
291	C.13.6	CRUDN behaviour	241
292	C.14	Security Profile	241
293	C.14.1	Introduction	241
294	C.14.2	Well-known URI.....	241
295	C.14.3	Resource type	241
296	C.14.4	OpenAPI 2.0 definition.....	241
297	C.14.5	Property definition	243
298	C.14.6	CRUDN behaviour	244
299	C.15	Token Refresh	244
300	C.15.1	Introduction	244
301	C.15.2	Well-known URI.....	244
302	C.15.3	Resource type	244
303	C.15.4	OpenAPI 2.0 definition.....	244
304	C.15.5	Property definition	246
305	C.15.6	CRUDN behaviour	247
306	Annex D (informative)	OID definitions	248
307	Annex E (informative)	Security considerations specific to Bridged Protocols	250
308	E.1	Security Considerations specific to the AllJoyn Protocol	250
309	E.2	Security Considerations specific to the Bluetooth LE Protocol.....	250
310	E.3	Security Considerations specific to the oneM2M Protocol	250
311	E.4	Security Considerations specific to the U+ Protocol	250
312	E.5	Security Considerations specific to the Z-Wave Protocol.....	251
313	E.6	Security Considerations specific to the Zigbee Protocol	252
314			

FIGURES

315	
316	Figure 1 – OCF Interaction..... 10
317	Figure 2 – OCF Layers 12
318	Figure 3 – OCF Security Enforcement Points 14
319	Figure 4 – Use case-1 showing simple ACL enforcement 17
320	Figure 5 – Use case 2: A policy for the requested Resource is missing 17
321	Figure 6 – Use case-3 showing AMS supported ACL 18
322	Figure 7 – Use case-4 showing dynamically obtained ACL from an AMS..... 19
323	Figure 8 – Example Resource definition with opaque Properties 20
324	Figure 9 – Property Level Access Control 20
325	Figure 10 – Onboarding Overview..... 22
326	Figure 11 – OCF Onboarding Process 24
327	Figure 12 – OCF's SRM Architecture 28
328	Figure 13 – Discover New Device Sequence 35
329	Figure 14 – A Just Works OTM 37
330	Figure 15 – Random PIN-based OTM 39
331	Figure 16 – Manufacturer Certificate Based OTM Sequence 42
332	Figure 17 – Vendor-specific Owner Transfer Sequence..... 44
333	Figure 18 – Establish Device Identity Flow..... 45
334	Figure 19 – Owner Credential Selection Provisioning Sequence 46
335	Figure 20 – Symmetric Owner Credential Provisioning Sequence 47
336	Figure 21 – Asymmetric Owner Credential Provisioning Sequence..... 48
337	Figure 22 – Configure Device Services 50
338	Figure 23 – Provision New Device for Peer to Peer Interaction Sequence..... 51
339	Figure 24 – Example of Client-directed provisioning..... 54
340	Figure 25 – Example of Server-directed provisioning using a single provisioning service 55
341	Figure 26 – Example of Server-directed provisioning involving multiple support services 58
342	Figure 27 – Device state model..... 60
343	Figure 28 – OBT Sanity Check Sequence in SRESET 63
344	Figure 29 – Client-directed Certificate Transfer 80
345	Figure 30 – Client-directed CRL Transfer..... 81
346	Figure 32 – Asserting a role with a certificate role credential. 85
347	Figure 33 – Device connection with OCF Cloud 88
348	Figure 34 – OCF Security Resources 98
349	Figure 35 – /oic/sec/cred Resource and Properties 99
350	Figure 36 – /oic/sec/acl2 Resource and Properties 99
351	Figure 37 – /oic/sec/amacl Resource and Properties..... 100
352	Figure 38 – /oic/sec/sacl Resource and Properties..... 100
353	Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states 139

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

363 **Tables**

364	Table 1 – Discover New Device Details.....	35
365	Table 2 – A Just Works OTM Details.....	37
366	Table 3 – Random PIN-based OTM Details.....	39
367	Table 4 – Manufacturer Certificate Based OTM Details	43
368	Table 5 – Vendor-specific Owner Transfer Details	44
369	Table 6 – Establish Device Identity Details	45
370	Table 7 – Owner Credential Selection Details	46
371	Table 8 – Symmetric Owner Credential Assignment Details	47
372	Table 9 – Asymmetric Owner Credential Assignment Details	48
373	Table 10 – Configure Device Services Detail	50
374	Table 11 – Provision New Device for Peer to Peer Details	51
375	Table 12 – Steps describing Client -directed provisioning	54
376	Table 13 – Steps for Server-directed provisioning using a single provisioning service.....	55
377	Table 14 – Steps for Server-directed provisioning involving multiple support services.....	58
378	Table 15 – Mapping of Properties of the oic.r.account and oic.r.coapcloudconf Resources ...	59
379	Table 16 – X.509 v1 fields for Root CA Certificates.....	69
380	Table 17 - X.509 v3 extensions for Root CA Certificates	70
381	Table 18 - X.509 v1 fields for Intermediate CA Certificates	70
382	Table 19 – X.509 v3 extensions for Intermediate CA Certificates	71
383	Table 20 – X.509 v1 fields for End-Entity Certificates.....	71
384	Table 21 – X.509 v3 extensions for End-Entity Certificates	72
385	Table 22 – Device connection with the OCF Cloud flow	88
386	Table 23 – ACE2 Wildcard Matching Strings Description.....	94
387	Table 24 – Definition of the /oic/sec/doxm Resource.....	101
388	Table 25 – Properties of the /oic/sec/doxm Resource.....	101
389	Table 26 – Properties of the /oic/sec/didtype Property	102
390	Table 27 – Properties of the oic.sec.doxmtype Property.....	104
391	Table 28 – Definition of the oic.r.cred Resource.....	105
392	Table 29 – Properties of the /oic/sec/cred Resource	106
393	Table 30 – Properties of the oic.sec.cred Property.....	107
394	Table 31: Properties of the oic.sec.credusagetype Property.....	108
395	Table 32 – Properties of the oic.sec.pubdatatype Property.....	108
396	Table 33 – Properties of the oic.sec.privdatatype Property.....	108
397	Table 34 – Properties of the oic.sec.optdatatype Property	109
398	Table 35 – Definition of the oic.sec.roletype Property.	109
399	Table 36 – Value Definition of the oic.sec.crmtype Property.....	111
400	Table 37 – 128-bit symmetric key	112
401	Table 38 – 256-bit symmetric key	112

402	Table 39 – Definition of the oic.r.crl Resource.....	114
403	Table 40 – Properties of the oic.r.crl Resource	114
404	Table 41 – BNF Definition of OCF ACL	114
405	Table 42 – Definition of the oic.r.acl Resource	117
406	Table 43 – Properties of the oic.r.acl Resource.....	117
407	Table 44 – Properties of the oic.r.ace Property	118
408	Table 45 – Value Definition of the oic.sec.crudntype Property.....	118
409	Table 46 – Definition of the oic.sec.acl2 Resource	119
410	Table 47 – Properties of the oic.sec.acl2 Resource.....	119
411	Table 48 – oic.sec.ace2 data type definition.	120
412	Table 49 – oic.sec.ace2.resource-ref data type definition.....	120
413	Table 50 – Value definition oic.sec.conntype Property	120
414	Table 51 – Definition of the oic.r.amacl Resource	123
415	Table 52 – Properties of the oic.r.amacl Resource	123
416	Table 53 – Definition of the oic.r.sacl Resource	123
417	Table 54 – Properties of the oic.r.sacl Resource	123
418	Table 55 – Properties of the oic.sec.sigtype Property.....	124
419	Table 56 – Definition of the oic.r.pstat Resource	124
420	Table 57 – Properties of the oic.r.pstat Resource.....	125
421	Table 58 – Properties of the /oic/sec/dostype Property	126
422	Table 59 – Definition of the oic.sec.dpmtype Property.....	128
423	Table 60 – Value Definition of the oic.sec.dpmtype Property (Low-Byte).....	128
424	Table 61 – Value Definition of the oic.sec.dpmtype Property (High-Byte)	128
425	Table 62 – Definition of the oic.sec.pomtype Property.....	129
426	Table 63 – Value Definition of the oic.sec.pomtype Property.....	129
427	Table 64 – Definition of the oic.r.csr Resource	129
428	Table 65 – Properties of the oic.r.csr Resource.....	129
429	Table 66 – Definition of the oic.r.roles Resource.....	131
430	Table 67 – Properties of the oic.r.roles Resource.....	131
431	Table 68 – Definition of the oic.r.account Resource	132
432	Table 69 – Properties of the oic.r.account Resource	132
433	Table 70 – Definition of the oic.r.session Resource.....	133
434	Table 71 – Properties of the oic.r.session Resource	133
435	Table 72 – Definition of the oic.r.tokenrefresh Resource	134
436	Table 73 – Properties of the oic.r.tokenrefresh Resource	135
437	Table 74 – Core Resource Properties Access Modes given various Device States	137
438	Table 75 – Examples of Sensitive Data.....	143
439	Table 76 – Definition of the oic.sec.sp Resource.....	152
440	Table 77 – Properties of the oic.sec.sp Resource	152

441	Table B.1 – OCF Security Profile	162
442	Table C.1 – Alphabetized list of security resources	163
443	Table C.2 – The Property definitions of the Resource with type 'rt' = ['oic.r.account']	166
444	Table C.3 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.account']	167
445	Table C.4 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl']	174
446	Table C.5 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl']	174
447	Table C.6 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl2']	184
448	Table C.7 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl2']	185
449	Table C.8 – The Property definitions of the Resource with type 'rt' = ['oic.r.amacl']	188
450	Table C.9 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.amacl']	189
451	Table C.10 – The Property definitions of the Resource with type 'rt' = ['oic.r.cred']	199
452	Table C.11 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.cred']	200
453	Table C.12 – The Property definitions of the Resource with type 'rt' = ['oic.r.crl']	202
454	Table C.13 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.crl']	203
455	Table C.14 – The Property definitions of the Resource with type 'rt' = ['oic.r.csr']	205
456	Table C.15 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.csr']	205
457	Table C.16 – The Property definitions of the Resource with type 'rt' = ['oic.r.doxxm']	209
458	Table C.17 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.doxxm']	211
459	Table C.18 – The Property definitions of the Resource with type 'rt' = ['oic.r.pstat']	215
460	Table C.19 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.pstat']	220
461	Table C.20 – The Property definitions of the Resource with type 'rt' = ['oic.r.roles']	229
462	Table C.21 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.roles']	229
463	Table C.22 – The Property definitions of the Resource with type 'rt' = ['oic.r.sacl']	237
464	Table C.23 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sacl']	238
465	Table C.24 – The Property definitions of the Resource with type 'rt' = ['oic.r.session']	240
466	Table C.25 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.session']	241
467	Table C.26 – The Property definitions of the Resource with type 'rt' = ['oic.r.sp']	243
468	Table C.27 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sp']	244
469	Table C.28 – The Property definitions of the Resource with type 'rt' = ['oic.r.tokenrefresh'] ..	246
470	Table C.29 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.tokenrefresh'] ..	247
471	Table E.1 GAP security mode	250
472	Table E.2 TLS 1.2 Cipher Suites used by U+	251
473	Table E.3 Z-Wave Security Class	252
474	Table E.4 Zigbee 3.0 Security Levels to the Network, and Application Support layers	253
475		
476		

477 **1 Scope**

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

482 **2 Normative References**

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

487 ISO/IEC 30118-1:2018 Information technology -- Open Connectivity Foundation (OCF)
488 Specification -- Part 1: Core specification
489 <https://www.iso.org/standard/53238.html>
490 Latest version available at:
491 https://openconnectivity.org/specs/OCF_Core_Specification.pdf

492 ISO/IEC 30118-3:2018 Information technology -- Open Connectivity Foundation (OCF)
493 Specification -- Part 3: Bridging specification
494 <https://www.iso.org/standard/74240.html>
495 Latest version available at:
496 https://openconnectivity.org/specs/OCF_Bridging_Specification.pdf

497 ISO/IEC 30118-4:2018 Information technology -- Open Connectivity Foundation (OCF)
498 Specification -- Part 4: Resource type specification
499 <https://www.iso.org/standard/74241.html>
500 Latest version available at:
501 https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping.pdf

502 OCF Wi-Fi Easy Setup, *Open Connectivity Foundation Wi-Fi Easy Setup*, Version 2.0.1
503 Latest version available at:
504 https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification.pdf

505 OCF Cloud Specification, *Open Connectivity Foundation Cloud*, Version 2.0.1
506 Latest version available at:
507 https://openconnectivity.org/specs/OCF_Cloud_Specification.pdf

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

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

511 IETF RFC 2898, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, September
512 2000, <https://tools.ietf.org/html/rfc2898>

513 IETF RFC 2986, *PKCS #10: Certification Request Syntax Specification Version 1.7*, November
514 2000, <https://tools.ietf.org/html/rfc2986>

515 IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December
516 2005, <https://tools.ietf.org/html/rfc4279>

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

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

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

523 IETF RFC 5489, *ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)*, March 2009,
524 <https://tools.ietf.org/html/rfc5489>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

551

552 **3 Terms, definitions, and abbreviated terms**

553 **3.1 Terms and definitions**

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

556 ISO and IEC maintain terminological databases for use in standardization at the following
557 addresses:

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

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

560 **3.1.1**

561 **Access Management Service (AMS)**

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

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

565 **3.1.2**

566 **Access Token**

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

569 **3.1.3**

570 **Authorization Provider**

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

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

574 **3.1.4**

575 **Client**

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

577 **3.1.5**

578 **Credential Management Service (CMS)**

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

581 **3.1.6**

582 **Device**

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

584 **3.1.7**

585 **Device Class**

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

589 **3.1.8**

590 **Device ID**

591 a stack instance identifier.

592 **3.1.9**

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

594 a logical entity that establishes device ownership

595 **3.1.10**
596 **Device Registration**
597 a process by which Device is enrolled/registered to the OCF Cloud infrastructure (using Device
598 certificate and unique credential) and becomes ready for further remote operation through the
599 cloud interface (e.g. connection to remote Resources or publishing of its own Resources for
600 access).

601 **3.1.11**
602 **End-Entity**
603 any certificate holder which is not a Root or Intermediate Certificate Authority.

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

605 **3.1.12**
606 **Entity**

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

608 **3.1.13**
609 **OCF Interface**

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

611 **3.1.14**
612 **Intermediary**

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

615 **3.1.15**
616 **OCF Cipher Suite**

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

620 **3.1.16**
621 **OCF Cloud User**

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

624 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
625 organization that onboarded that Device. The OCF Cloud User delegates, to the OCF Cloud authority, authority to route
626 between Devices registered by the OCF Cloud User. The OCF Cloud delegates, to the OCF Cloud User, authority to
627 select the set of Devices which can register and use the services of the OCF Cloud.

628 **3.1.17**
629 **OCF Rooted Certificate Chain**

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

633 **3.1.18**
634 **Onboarding Tool (OBT)**

635 a tool that implements DOTS(3.1.9), AMS(3.1.1) and CMS(3.1.5) functionality

636 **3.1.19**
637 **Out of Band Method**

638 any mechanism for delivery of a secret from one party to another, not specified by OCF

639 **3.1.20**
640 **Owner Credential (OC)**

641 Credential, provisioned by an Onboarding Tool to a Device during onboarding, for the purposes
642 of mutual authentication of the Device and Onboarding Tool during subsequent interactions

643 **3.1.21**
644 **Platform ID**
645 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

646 **3.1.22**
647 **Property**
648 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

649 **3.1.23**
650 **Resource**
651 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

652 **3.1.24**
653 **Role (Network context)**
654 stereotyped behavior of a Device; one of [Client, Server or Intermediary]

655 **3.1.25**
656 **Role Identifier**
657 a Property of an OCF credentials Resource or element in a role certificate that identifies a
658 privileged role that a Server Device associates with a Client Device for the purposes of making
659 authorization decisions when the Client Device requests access to Device Resources.

660 **3.1.26**
661 **Secure Resource Manager (SRM)**
662 a module in the OCF Core that implements security functionality that includes management of
663 security Resources such as ACLs, credentials and Device owner transfer state.

664 **3.1.27**
665 **Security Virtual Resource (SVR)**
666 a resource supporting security features.
667 Note 1 to entry: For a list of all the SVRs please see Clause 13.

668 **3.1.28**
669 **Server**
670 Note 1 to entry: The details are defined in ISO/IEC 30118-1:2018.

671 **3.1.29**
672 **Trust Anchor**
673 a well-defined, shared authority, within a trust hierarchy, by which two cryptographic entities (e.g.
674 a Device and an onboarding tool) can assume trust

675 **3.1.30**
676 **Unique Authenticable Identifier**
677 a unique identifier created from the hash of a public key and associated OCF Cipher Suite that is
678 used to create the Device ID.
679 Note 1 to entry: The ownership of a UAID may be authenticated by peer Devices.

680 **3.1.31**
681 **Device Configuration Resource (DCR)**
682 a Resource that is any of the following:
683 a) a Discovery Core Resource, or
684 b) a Security Virtual Resource, or
685 c) a WiFi Easy Setup Resource, or
686 d) a CoAP Cloud Conf Resource.

687 **3.1.32**
688 **Non-Configuration Resource (NCR)**
689 a Resource that is not a Device Configuration Resource (3.1.31).

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

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

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

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

707 **3.1.38**
708 **OCF Security Domain**
709 set of onboarded OCF Devices that are provisioned with credentialing information for confidential
710 communication with one another

711 **3.2 Abbreviated terms**

712 **3.2.1**
713 **AC**
714 Access Control

715 **3.2.2**
716 **ACE**
717 Access Control Entry

718 **3.2.3**
719 **ACL**
720 Access Control List

721 **3.2.4**
722 **AES**
723 Advanced Encryption Standard

724 Note 1 to entry: See NIST FIPS 197, "Advanced Encryption Standard (AES)"

725 **3.2.5**
726 **AMS**
727 Access Management Service

728 **3.2.6**
729 **CMS**
730 Credential Management Service

731 **3.2.7**
732 **CRUDN**
733 CREATE, RETREIVE, UPDATE, DELETE, NOTIFY

734 **3.2.8**
735 **CSR**
736 Certificate Signing Request

737 **3.2.9**
738 **CVC**
739 Code Verification Certificate

740 **3.2.10**
741 **ECC**
742 Elliptic Curve Cryptography

743 **3.2.11**
744 **ECDSA**
745 Elliptic Curve Digital Signature Algorithm

746 **3.2.12**
747 **EKU**
748 Extended Key Usage

749 **3.2.13**
750 **EPC**
751 Embedded Platform Credential

752 **3.2.14**
753 **EPK**
754 Embedded Public Key

755 **3.2.15**
756 **DOTS**
757 Device Ownership Transfer Service

758 **3.2.16**
759 **DPKP**
760 Dynamic Public Key Pair

761 **3.2.17**
762 **ID**
763 Identity/Identifier

764 **3.2.18**
765 **JSON**
766 See ISO/IEC 30118-1:2018.

767 **3.2.19**
768 **JWS**
769 JSON Web Signature.

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

771 **3.2.20**
772 **KDF**
773 Key Derivation Function

774 **3.2.21**
775 **MAC**
776 Message Authentication Code

777 **3.2.22**
778 **MITM**
779 Man-in-the-Middle

780 **3.2.23**
781 **NVRAM**
782 Non-Volatile Random-Access Memory

783 **3.2.24**
784 **OC**
785 Owner Credential

786 **3.2.25**
787 **OCSP**
788 Online Certificate Status Protocol

789 **3.2.26**
790 **OBT**
791 Onboarding Tool

792 **3.2.27**
793 **OID**
794 Object Identifier

795 **3.2.28**
796 **OTM**
797 Owner Transfer Method

798 **3.2.29**
799 **OOB**
800 Out of Band

801 **3.2.30**
802 **OWASP**
803 Open Web Application Security Project. See <https://www.owasp.org/>

804 **3.2.31**
805 **PE**
806 Policy Engine

807 **3.2.32**
808 **PIN**
809 Personal Identification Number

810 **3.2.33**
811 **PPSK**
812 PIN-authenticated pre-shared key

813 **3.2.34**
814 **PRF**
815 Pseudo Random Function

816 **3.2.35**
817 **PSI**
818 Persistent Storage Interface

819 **3.2.36**
820 **PSK**
821 Pre Shared Key

822 **3.2.37**
823 **RBAC**
824 Role Based Access Control

825 **3.2.38**
826 **RM**
827 Resource Manager

828 **3.2.39**
829 **RNG**
830 Random Number Generator

831 **3.2.40**
832 **SACL**
833 Signed Access Control List

834 **3.2.41**
835 **SBAC**
836 Subject Based Access Control

837 **3.2.42**
838 **SEE**
839 Secure Execution Environment

840 **3.2.43**
841 **SRM**
842 Secure Resource Manager

843 **3.2.44**
844 **SVR**
845 Security Virtual Resource

846 **3.2.45**
847 **SW**
848 Software

849 **3.2.46**
850 **UAID**
851 Unique Authenticable Identifier

852 **3.2.47**
853 **URI**
854 See ISO/IEC 30118-1:2018.

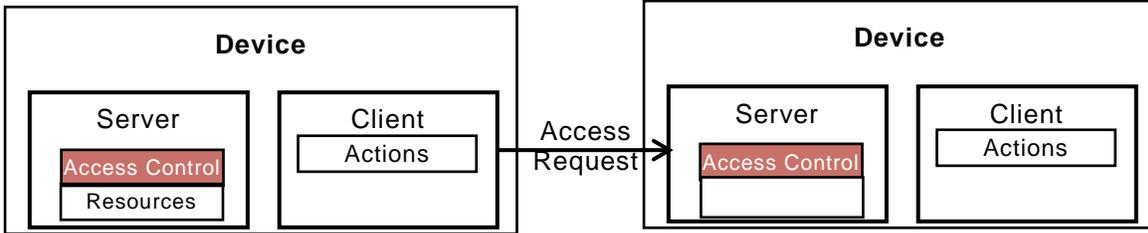
855 **4 Document Conventions and Organization**

856 **4.1 Conventions**

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

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

861 Figure 1 depicts interaction between OCF Devices.



862

863

Figure 1 – OCF Interaction

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

868 **4.2 Notation**

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

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

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

875 **Recommended (or should).**

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

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

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

886 **Conditionally allowed (CA)**

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

889 **Conditionally required (CR)**

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

893 **DEPRECATED**

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

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

901 Words that are emphasized are printed in italic.

902 **4.3 Data types**

903 See ISO/IEC 30118-1:2018.

904 **4.4 Document structure**

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

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

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

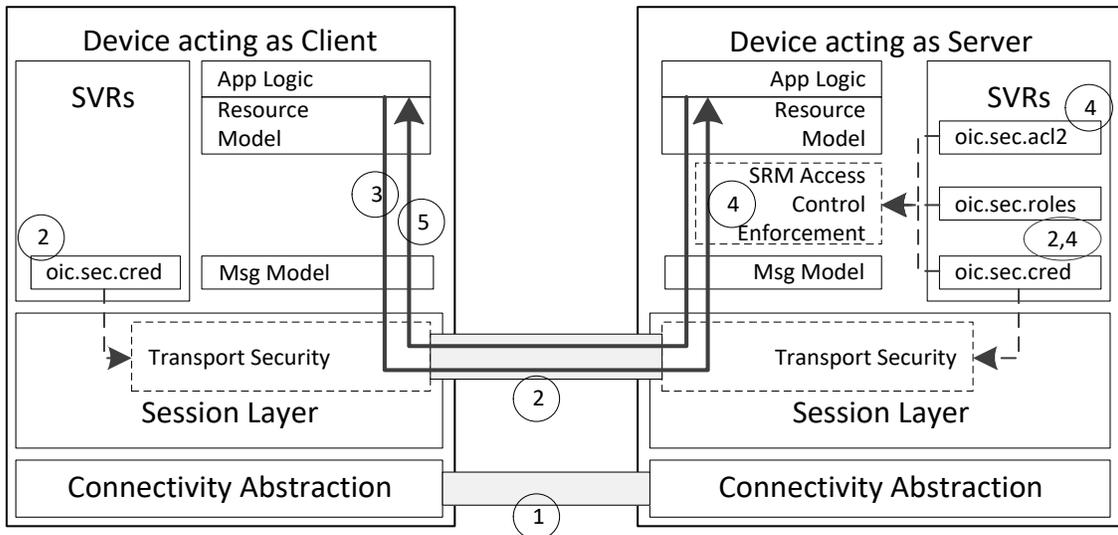
911

912 **5 Security Overview**

913 **5.1 Preamble**

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

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



923
924

925 **Figure 2 – OCF Layers**

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

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

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

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

944 4) The Server receives the request.

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

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

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

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

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

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

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

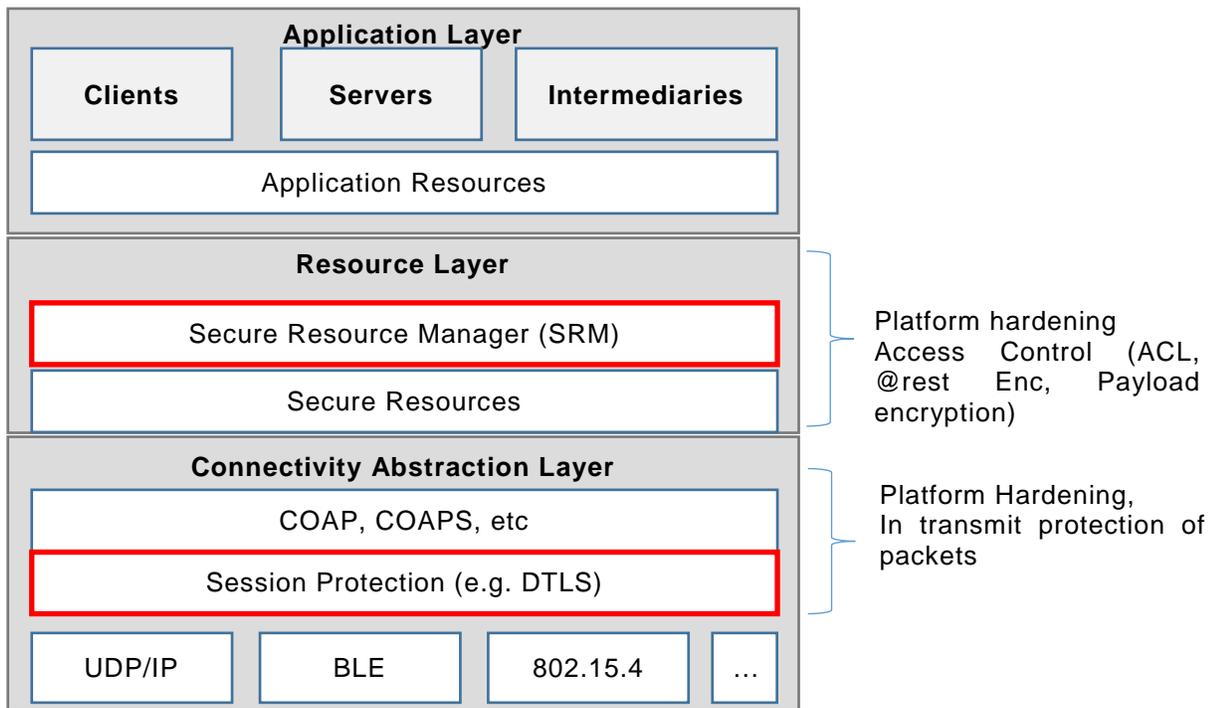
962 5) The Server sends a response back to the Client.

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

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

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

977 Figure 3 depicts OCF Security Enforcement Points.



978

979

980

Figure 3 – OCF Security Enforcement Points

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

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

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

991 **5.2 Access Control**

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

998 Implementation of access control relies on a-priori definition of a set of access policies for the
 999 Resource. The policies may be stored by a local ACL or an Access Management Service (AMS)
 1000 in form of Access Control Entries (ACE). Two types of access control mechanisms can be
 1001 applied:

1002 – Subject-based access control (SBAC), where each ACE will match a subject (e.g. identity of
1003 requestor) of the requesting entity against the subject included in the policy defined for
1004 Resource. Asserting the identity of the requestor requires an authentication process.

1005 – Role-based Access Control (RBAC), where each ACE will match a role identifier included in
1006 the policy for the Resource to a role identifier associated with the requestor

1007 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only
1008 local references and there is an ACE matching the Atomic Measurement Resource which permits
1009 the request, then the corresponding requests to linked Resources are permitted by the OCF
1010 Server. The present paragraph shall apply to any Resource Type based on the Atomic
1011 Measurement Resource Type.

1012 NOTE The definition of an Atomic Measurement Resource prohibits direct access to the linked Resources. The nature
1013 of an Atomic Measurement also prohibits updating the "links" to add or remove Resources. Consequently, there is no
1014 risk of privilege escalation when using the ACE of an Atomic Measurement Resource to govern access to its linked
1015 Resources.

1016 If an OCF Server receives a batch request to a Collection Resource containing only local
1017 references and there is an ACE matching the Collection Resource which permits the request,
1018 then the corresponding requests to linked Resources are permitted by the OCF Server. The
1019 present paragraph shall apply to any Resource Type based on the Collection Resource Type.

1020 NOTE This implies that the ACEs of the Collection Resource permit access to all the Collection's linked Resources via
1021 the batch OCF Interface, even if there are no ACEs permitting direct access to some or all the linked Resources. If not
1022 tightly governed, this could lead to privilege escalation. Restrictions on the use of Collection Resources have been
1023 provided in ISO/IEC 30118-1:2018 to mitigate the risk of privilege escalation. For example, ISO/IEC 30118-1:2018
1024 prohibits updating "links" of a Collection Resource with the intent of obtaining access to the added Resource according
1025 to the ACEs of the Collection, when access to the Resource would have otherwise been denied.

1026 In the OCF access control model, access to a Resource instance requires an associated access
1027 control policy. This means, each Device acting as Server, needs to have an ACE permitting
1028 access to each Resource it is protecting. This criterion can be satisfied for a Resource A if there
1029 is an ACE permitting batch requests to access Resource B containing a Link to Resource A, even
1030 if there are no ACEs permitting requests which access Resource A directly. Examples of the
1031 Resource Type for Resource B is the Atomic Measurement Resource Type and the Collection
1032 Resource Type. The lack of an ACE permitting access to a Resource, either directly or via a Link
1033 results in the Resource being inaccessible.

1034 The ACE only applies if the ACE matches both the subject (i.e. OCF Client) and the requested
1035 Resource. There are multiple ways a subject could be matched, (1) DeviceID, (2) Role Identifier
1036 or (3) wildcard. The way in which the client connects to the server may be relevant context for
1037 making access control decisions. Wildcard matching on authenticated vs. unauthenticated and
1038 encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject
1039 classes.

1040 Example Wildcard Matching Policy:

```
1041 "aclist2": [  
1042   {  
1043     "subject": {"conntype": "anon-clear" },  
1044     "resources": [  
1045       { "wc": "*" }  
1046     ],  
1047     "permission": 31  
1048   },  
1049   {  
1050     "subject": {"conntype": "auth-crypt" },  
1051     "resources": [  

```

```
1052     { "wc":"*" }
1053   ],
1054   "permission": 31
1055   },
1056 ]
```

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

1059 ACL Resource requires the same security protection as other sensitive Resources, when it comes
1060 to both storage and handling by SRM and PSI. Thus hardening of an underlying Platform (HW
1061 and SW) must be considered for protection of ACLs and as explained in clause 5.2.2 ACLs may
1062 have different scoping levels and thus hardening needs to be specially considered for each
1063 scoping level. For instance a physical device may host multiple Device implementations and thus
1064 secure storage, usage and isolation of ACLs for different Servers on the same Device needs to
1065 be considered.

1066 **5.2.1 ACL Architecture**

1067 **5.2.1.1 ACL Architecture General**

1068 The Server examines the Resource(s) requested by the client before processing the request. The
1069 access control resources (e.g. "/oic/sec/acl", "/oic/sec/acl2") are searched to find one or more
1070 ACE entries that match the requestor and the requested Resources. If a match is found then
1071 permission and period constraints are applied. If more than one match is found then the logical
1072 UNION of permissions is applied to the overlapping periods.

1073 The server uses the connection context to determine whether the subject has authenticated or
1074 not and whether data confidentiality has been applied or not. Subject matching wildcard policies
1075 can match on each aspect. If the user has authenticated, then subject matching may happen at
1076 increased granularity based on role or device identity.

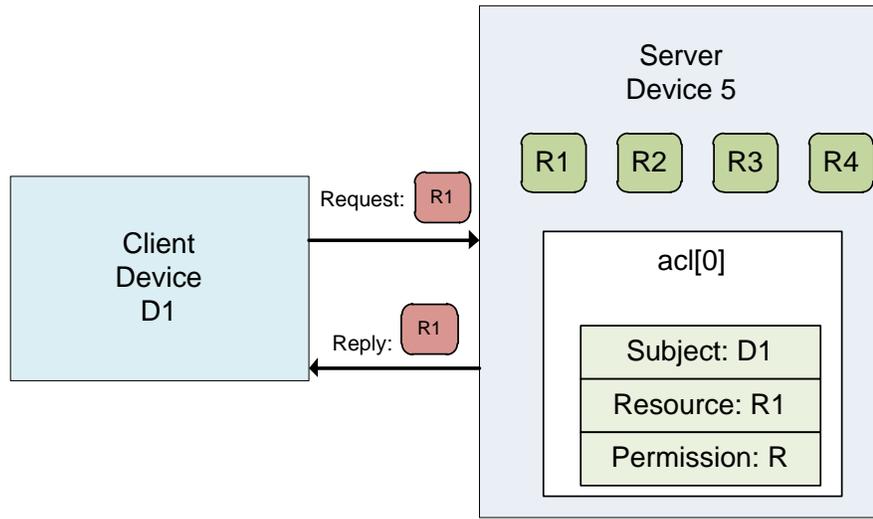
1077 Each ACE contains the permission set that will be applied for a given Resource requestor.
1078 Permissions consist of a combination of CREATE, RETREIVE, UPDATE, DELETE and NOTIFY
1079 (CRUDN) actions. Requestors authenticate as a Device and optionally operating with one or more
1080 roles. Devices may acquire elevated access permissions when asserting a role. For example, an
1081 ADMINISTRATOR role might expose additional Resources and OCF Interfaces not normally
1082 accessible.

1083 **5.2.1.2 Use of local ACLs**

1084 Servers may host ACL Resources locally. Local ACLs allow greater autonomy in access control
1085 processing than remote ACL processing by an AMS.

1086 The following use cases describe the operation of access control

1087 Use Case 1: As depicted in Figure 4, Server Device hosts 4 Resources (R1, R2, R3 and R4).
1088 Client Device D1 requests access to Resource R1 hosted at Server Device 5. ACL[0]
1089 corresponds to Resource R1 and includes D1 as an authorized subject. Thus, Device D1 receives
1090 access to Resource R1 because the local ACL /oic/sec/acl/0 matches the request.

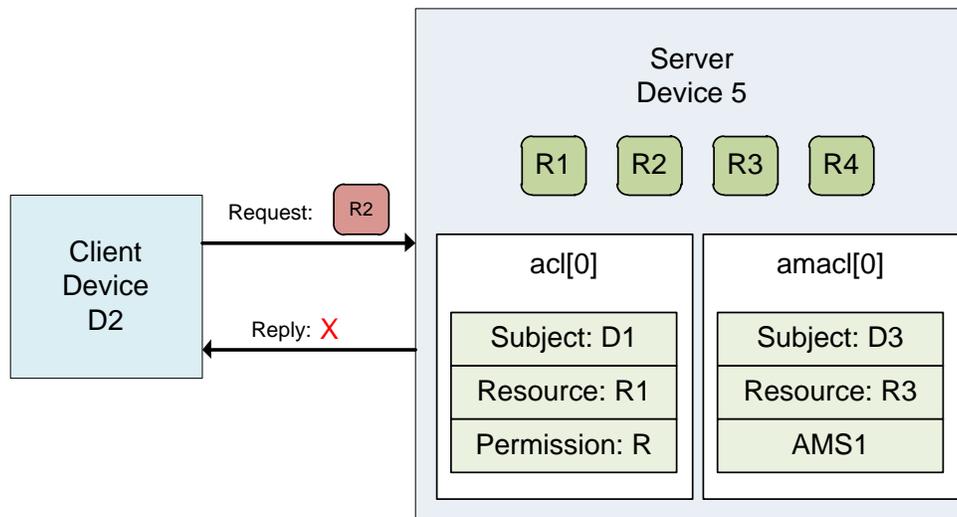


1091

1092

Figure 4 – Use case-1 showing simple ACL enforcement

1093 Use Case 2: As depicted in Figure 5, Client Device D2 access is denied because no local ACL
 1094 match is found for subject D2 pertaining Resource R2 and no AMS policy is found.



1095

1096

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

1097 **5.2.1.3 Use of AMS**

1098 AMS improves ACL policy management. However, they can become a central point of failure.
 1099 Due to network latency overhead, ACL processing may be slower through an AMS.

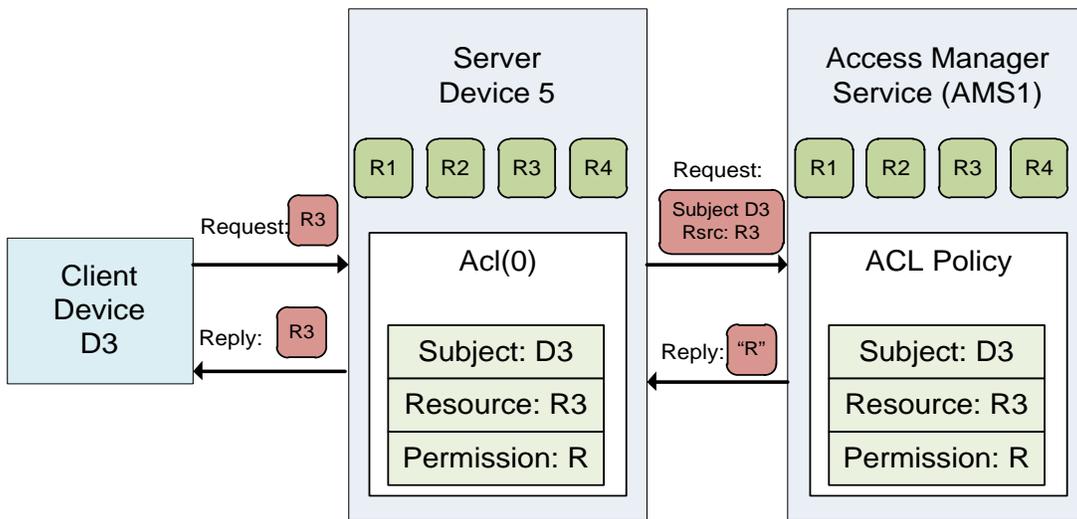
1100 AMS centralizes access control decisions, but Server Devices retain enforcement duties. The
 1101 Server shall determine which ACL mechanism to use for which Resource set. The /oic/sec/amacl
 1102 Resource is an ACL structure that specifies which Resources will use an AMS to resolve access
 1103 decisions. The /oic/sec/amacl may be used in concert with local ACLs ("/oic/sec/acl").

1104 The AMS is authenticated by referencing a credential issued to the device identifier contained in
 1105 "/oic/sec/acl2.rowneruuid".

1106 The Server Device may proactively open a connection to the AMS using the Device ID found in
 1107 /oic/sec/acl2.rowneruuid. Alternatively, the Server may reject the Resource access request with
 1108 an error, ACCESS_DENIED_REQUIRES_SACL that instructs the requestor to obtain a suitable
 1109 ACE policy using a SACL Resource /oic/sec/sacl. The /oic/sec/sacl signature may be validated
 1110 using the credential Resource associated with the /oic/sec/acl2.rowneruuid.

1111 The following use cases describe access control using the AMS:

1112 Use Case 3: As depicted in Figure 6, Device D3 requests and receives access to Resource R3
 1113 with permission Perm1 because the /oic/sec/amacl/0 matches a policy to consult the Access
 1114 Manager Server AMS1 service



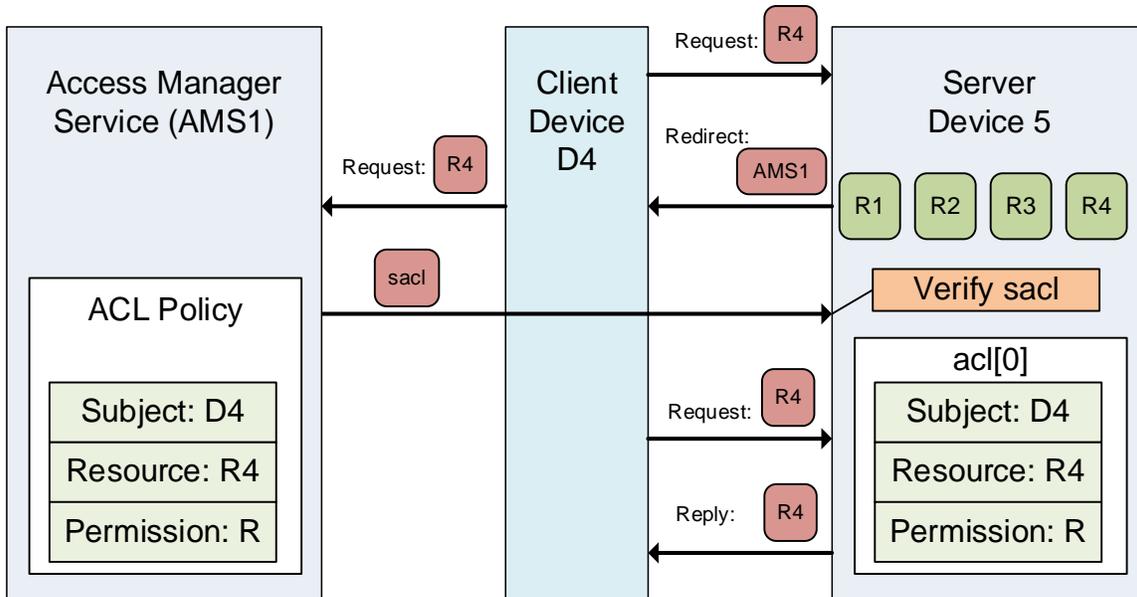
1115

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

1117 Use Case 4: As depicted in Figure 7, Client Device D4 requests access to Resource R4 from
 1118 Server Device 5, which fails to find a matching ACE and redirects the Client Device D4 to AMS1
 1119 by returning an error identifying AMS1 as a /oic/sec/sacl Resource issuer. Device D4 obtains
 1120 Sacl1 signed by AMS1 and forwards the SACL to Server D5. D5 verifies the signature in the
 1121 /oic/sec/sacl Resource and evaluates the ACE policy that grants Perm2 access.

1122 ACE redirection may occur when D4 receives an error result with reason code indicating no
 1123 match exists (i.e. ACCESS_DENIED_NO_ANCE). D4 reads the /oic/sec/acl2 Resource to find the
 1124 rowneruuid which identifies the AMS and then submits a request to be provisioned, in this
 1125 example the AMS chooses to supply a SACL Resource, however it may choose to re-provision
 1126 the local ACL Resources /oic/sec/acl and /oic/sec/acl2. The request is reissued subsequently. D4
 1127 is presumed to have been introduced to the AMS as part of Device onboarding or through
 1128 subsequent credential provisioning actions.

1129 If not, a Credential Management Service (CMS) can be consulted to provision needed credentials.



1130

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

1132 5.2.2 Access Control Scoping Levels

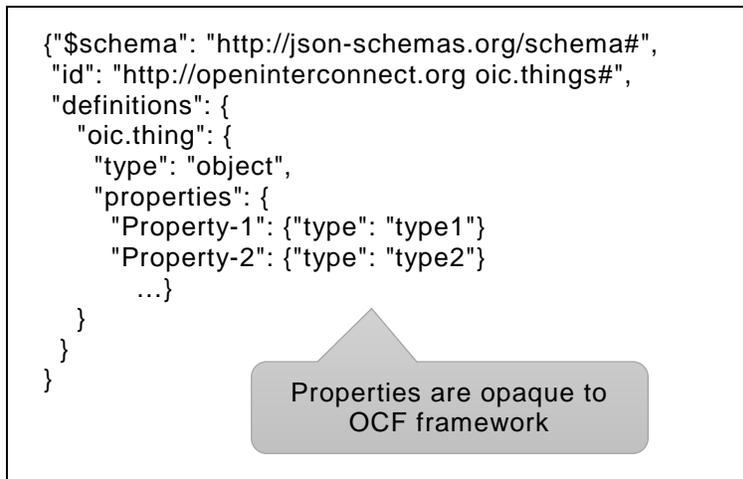
1133 **Group Level Access** - Group scope means applying AC to the group of Devices that are grouped
1134 for a specific context. Group Level Access means all group members have access to group data
1135 but non-group members must be granted explicit access. Group level access is implemented
1136 using Role Credentials and/or connection type

1137 **OCF Device Level Access** – OCF Device scope means applying AC to an individual Device,
1138 which may contain multiple Resources. Device level access implies accessibility extends to all
1139 Resources available to the Device identified by Device ID. Credentials used for AC mechanisms
1140 at Device are OCF Device-specific.

1141 **OCF Resource Level Access** – OCF Resource level scope means applying AC to individual
1142 Resources. Resource access requires an ACL that specifies how the entity holding the Resource
1143 (Server) shall make a decision on allowing a requesting entity (Client) to access the Resource.

1144 **Property Level Access** - Property level scope means applying AC only to an individual Property
1145 Property level access control is only achieved by creating a Resource that contains a single
1146 Property.

1147 Controlling access to static Resources where it is impractical to redesign the Resource, it may
1148 appropriate to introduce a collection Resource that references the child Resources having
1149 separate access permissions. An example is shown Figure 8, where an "oic.thing" Resource has
1150 two properties: Property-1 and Property-2 that would require different permissions.

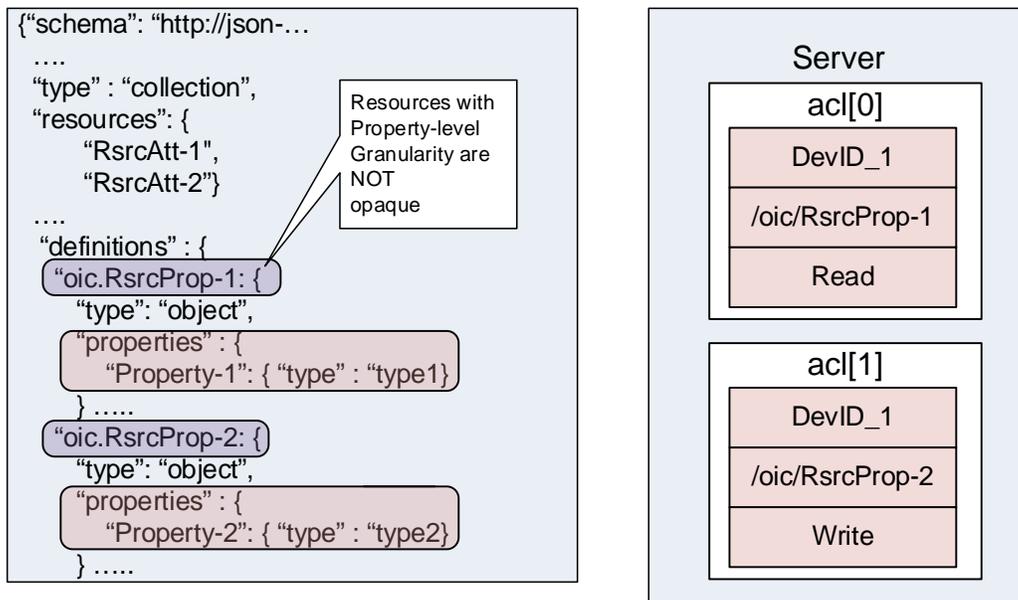


1151

1152

Figure 8 – Example Resource definition with opaque Properties

1153 Currently, OCF framework treats property level information as opaque; therefore, different
 1154 permissions cannot be assigned as part of an ACL policy (e.g. read-only permission to Property-1
 1155 and write-only permission to Property-2). Thus, as shown in Figure 9, the "oic.thing" is split into
 1156 two new Resource "oic.RsrcProp-1" and "oic.RsrcProp-2". This way, Property level ACL can be
 1157 achieved through use of Resource-level ACLs.



1158

1159

Figure 9 – Property Level Access Control

1160 **5.3 Onboarding Overview**

1161 **5.3.1 Onboarding General**

1162 Before a Device becomes operational in an OCF environment and is able to interact with other
1163 Devices, it needs to be appropriately onboarded. The first step in onboarding a Device is to
1164 configure the ownership where the legitimate user that owns/purchases the Device uses an
1165 Onboarding tool (OBT) and using the OBT uses one of the Owner Transfer Methods (OTMs) to
1166 establish ownership. Once ownership is established, the OBT becomes the mechanism through
1167 which the Device can then be provisioned, at the end of which the Device becomes operational
1168 and is able to interact with other Devices in an OCF environment. An OBT shall be hosted on an
1169 OCF Device.

1170 Figure 10 depicts Onboarding Overview.

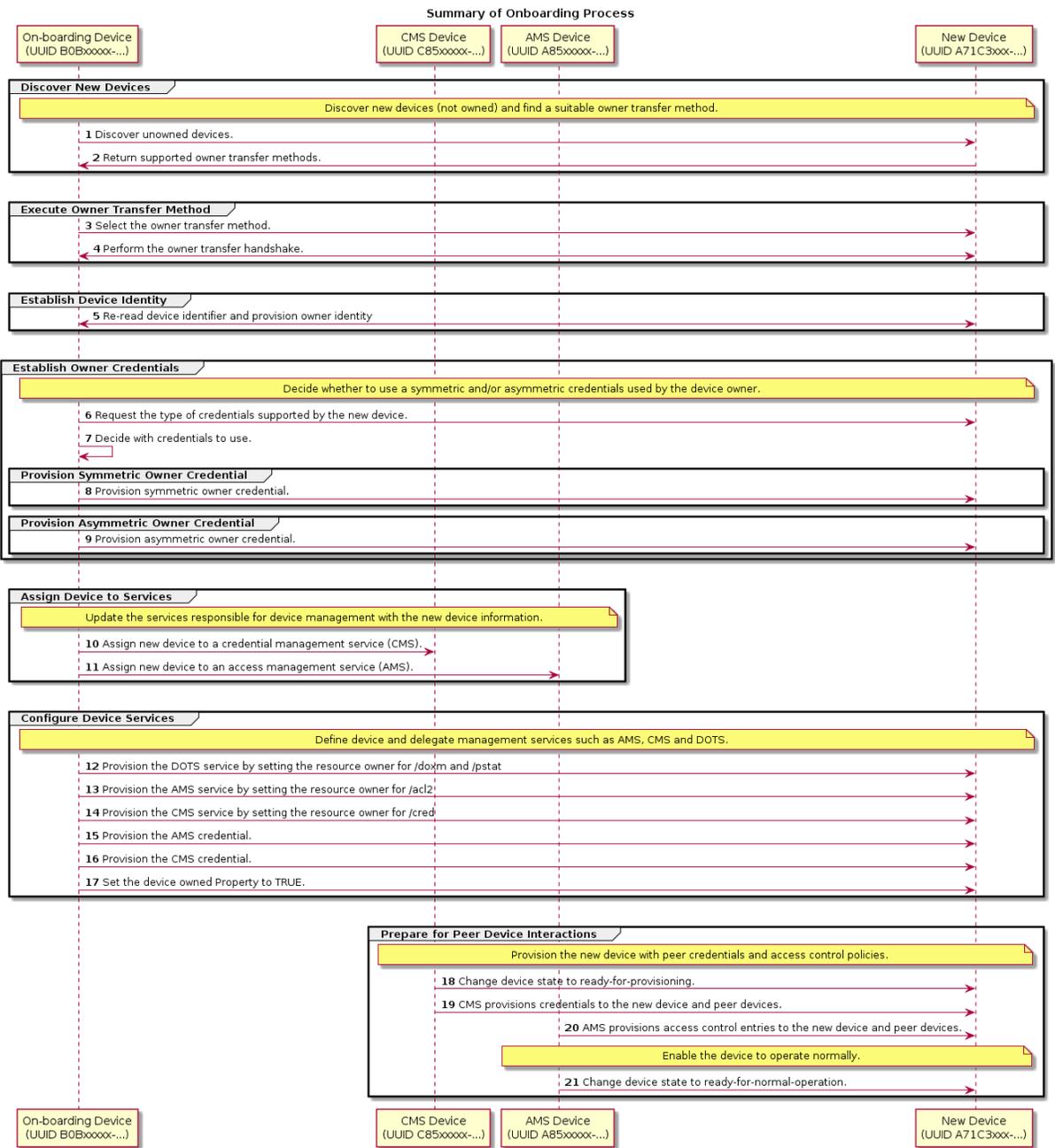


Figure 10 – Onboarding Overview

1171
1172

1173 This clause explains the onboarding and security provisioning process but leaves the provisioning
 1174 of non-security aspects to other OCF documents. In the context of security, all Devices are
 1175 required to be provisioned with minimal security configuration that allows the Device to securely
 1176 interact/communicate with other Devices in an OCF environment. This minimal security
 1177 configuration is defined as the Onboarded Device "Ready for Normal Operation" and is specified
 1178 in 7.5.

1179 Onboarding and provisioning implementations could utilize services defined outside this
 1180 document, it is expected that in using other services, trust between the device being onboarded
 1181 and the various tools is not transitive. This implies that the device being onboarded will
 1182 individually authenticate the credentials of each and every tool used during the onboarding

1183 process; that the tools not share credentials or imply a trust relationship where one has not been
1184 established.

1185 **5.3.2 OnBoarding Steps**

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

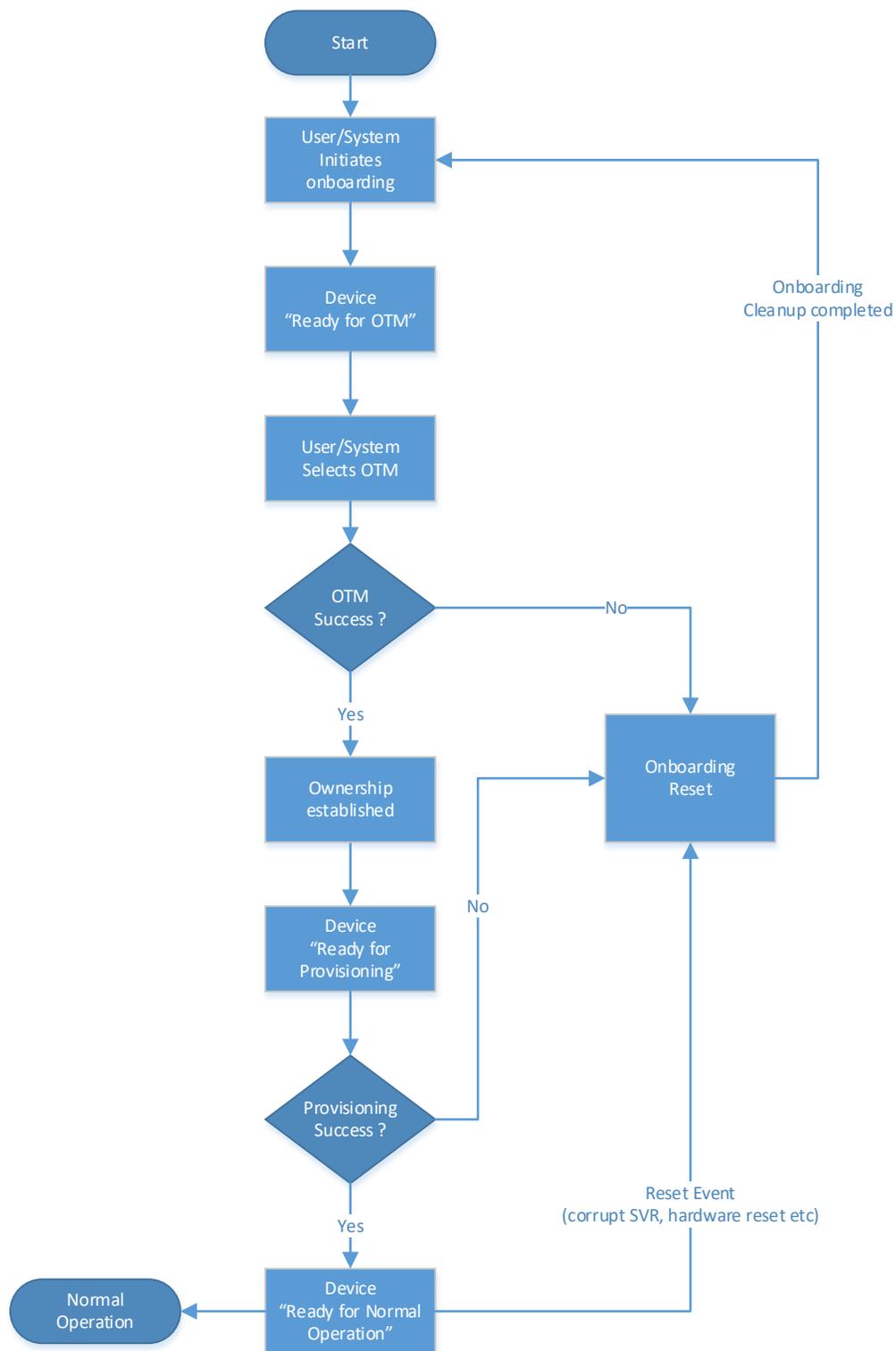


Figure 11 – OCF Onboarding Process

1193

1194

1195 5.3.3 Establishing a Device Owner

1196 The objective behind establishing Device ownership is to allow the legitimate user that
 1197 owns/purchased the Device to assert itself as the owner and manager of the Device. This is done
 Copyright Open Connectivity Foundation, Inc. © 2016-2019. All rights Reserved

1198 through the use of an OBT that includes the creation of an ownership context between the new
1199 Device and the OBT tool and asserts operational control and management of the Device. The
1200 OBT can be considered a logical entity hosted by tools/ Servers such as a network management
1201 console, a device management tool, a network-authoring tool, a network provisioning tool, a
1202 home gateway device, or a home automation controller. A physical device hosting the OBT will be
1203 subject to some security hardening requirements, thus preserving integrity and confidentiality of
1204 any credentials being stored. The tool/Server that establishes Device ownership is referred to as
1205 the OBT.

1206 The OBT uses one of the OTMs specified in 7.3 to securely establish Device ownership. The term
1207 owner transfer is used since it is assumed that even for a new Device, the ownership is
1208 transferred from the manufacturer/provider of the Device to the buyer/legitimate user of the new
1209 Device.

1210 An OTM establishes a new owner (the operator of OBT) that is authorized to manage the Device.
1211 Owner transfer establishes the following

- 1212 – The DOTS provisions an Owner Credential (OC) to the creds Property in the "/oic/sec/cred"
1213 Resource of the Device. This OC allows the Device and DOTS to mutually authenticate during
1214 subsequent interactions. The OC associates the DOTS DeviceID with the rowneruuid property
1215 of the "/oic/sec/doxm" resource establishing it as the resource owner. The DOTS records the
1216 identity of Device as part of ownership transfer.
- 1217 – The Device owner establishes trust in the Device through the OTM.
- 1218 – Preparing the Device for provisioning by providing credentials that may be needed.

1219 **5.3.4 Provisioning for Normal Operation**

1220 Once the Device has the necessary information to initiate provisioning, the next step is to
1221 provision additional security configuration that allows the Device to become operational. This can
1222 include setting various parameters and may also involve multiple steps. Also provisioning of
1223 ACL's for the various Resources hosted by the Server on the Device is done at this time. The
1224 provisioning step is not limited to this stage only. Device provisioning can happen at multiple
1225 stages in the Device's operational lifecycle. However specific security related provisioning of
1226 Resource and Property state would likely happen at this stage at the end of which, each Device
1227 reaches the Onboarded Device "Ready for Normal Operation" State. The "Ready for Normal
1228 Operation" State is expected to be consistent and well defined regardless of the specific OTM
1229 used or regardless of the variability in what gets provisioned. However individual OTM
1230 mechanisms and provisioning steps may specify additional configuration of Resources and
1231 Property states. The minimal mandatory configuration required for a Device to be in "Ready for
1232 Normal Operation" state is specified in 8.

1233 **5.3.5 Device Provisioning for OCF Cloud and Device Registration Overview**

1234 As mentioned in the start of Clause 5, communication between a Device and OCF Cloud is
1235 subject to different criteria in comparison to Devices which are within a single local network. The
1236 Device is configured in order to connect to the OCF Cloud by a Mediator as specified in the
1237 CoAPCloudConf Resource clauses in ISO/IEC 30118-8:2018. Provisioning includes the remote
1238 connectivity and local details such as URL where the OCF Cloud hosting environment can be
1239 found and the OCF Cloud verifiable Access Token.

1240 **5.3.6 OCF Compliance Management System**

1241 The OCF Compliance Management System (OCMS) is a service maintained by the OCF that
1242 provides Certification status and information for OCF Devices.

1243 The OCMS shall provide a JSON-formatted Certified Product List (CPL), hosted at the URI:
1244 <https://www.openconnectivity.org/certification/ocms-cpl.json>

1245 The OBT shall possess the Root Certificate needed to enable https connection to the URI
1246 <https://www.openconnectivity.org/certification/ocms-cpl.json>.

1247 The OBT should periodically refresh its copy of the CPL via the URI
1248 <https://www.openconnectivity.org/certification/ocms-cpl.json>, as appropriate to OCF Security
1249 Domain owner policy requirements.

1250 **5.4 Provisioning**

1251 **5.4.1 Provisioning General**

1252 In general, provisioning may include processes during manufacturing and distribution of the
1253 Device as well as processes after the Device has been brought into its intended environment
1254 (parts of onboarding process). In this document, security provisioning includes, processes after
1255 ownership transfer (even though some activities during ownership transfer and onboarding may
1256 lead to provisioning of some data in the Device) configuration of credentials for interacting with
1257 provisioning services, configuration of any security related Resources and credentials for dealing
1258 with any services that the Device need to contact later on.

1259 Once the ownership transfer is complete, the Device needs to engage with the CMS and AMS to
1260 be provisioned with proper security credentials and parameters for regular operation. These
1261 parameters can include:

- 1262 – Security credentials through a CMS, currently assumed to be deployed in the same OBT.
- 1263 – Access control policies and ACLs through an AMS, currently assumed to be deployed in the
1264 same OBT, but may be part of AMS in future.

1265 As mentioned, to accommodate a scalable and modular design, these functions are considered
1266 as services that in future could be deployed as separate servers. Currently, the deployment
1267 assumes that these services are all deployed as part of a OBT. Regardless of physical
1268 deployment scenario, the same security-hardening requirement) applies to any physical server
1269 that hosts the tools and security provisioning services discussed here.

1270 Devices are *aware* of their security provisioning status. Self-awareness allows them to be
1271 proactive about provisioning or re-provisioning security Resources as needed to achieve the
1272 devices operational goals.

1273 **5.4.2 Provisioning other services**

1274 To be able to support the use of potentially different device management service hosts, each
1275 Device Secure Virtual Resource (SVR) has an associated Resource owner identified in the
1276 Resource's rowneruuid Property.

1277 The DOTS shall update the rowneruuid Property of the /oic/sec/doxm and /oic/sec/pstat
1278 resources with the DOTS resource owner identifier.

1279 The DOTS shall update the rowneruuid Property of the /oic/sec/cred resource with the CMS
1280 resource owner identifier.

1281 The DOTS shall update the rowneruuid Property of the /oic/sec/acl2 resource with the AMS
1282 resource owner identifier

1283 When these OCF Services are configured, the Device may proactively request provisioning and
1284 verify provisioning requests are authorized. The DOTS shall provision credentials that enable
1285 secure connections between OCF Services and the new Device. The DOTS may initiate client-
1286 directed provisioning by signaling the OCF Service. The DOTS may initiate server-directed
1287 provisioning by setting tm Property of the /oic/sec/pstat Resource.

1288 **5.4.3 Provisioning Credentials for Normal Operation**

1289 The "/oic/sec/cred" Resource supports multiple types of credentials including:

- 1290 – Pairwise symmetric keys
- 1291 – Group symmetric keys
- 1292 – Certificates
- 1293 – Raw asymmetric keys

1294 The CMS shall securely provision credentials for Device-to-Device interactions using the CMS
1295 credential provisioned by the DOTS.

1296 The following example describes how a Device updates a symmetric key credential involving a
1297 peer Device. The Device discovers the credential to be updated; for example, a secure
1298 connection attempt fails. The Device requests its CMS to supply the updated credential. The
1299 CMS returns an updated symmetric key credential. The CMS updates the corresponding
1300 symmetric key credential on the peer Device.

1301 **5.4.4 Role Assignment and Provisioning for Normal Operation**

1302 The Servers, receiving requests for Resources they host, need to verify the role identifier(s)
1303 asserted by the Client requesting the Resource and compare that role identifier(s) with the
1304 constraints described in the Server's ACLs. Thus, a Client Device may need to be provisioned
1305 with one or more role credentials.

1306 Each Device holds the role information as a Property within the credential Resource.

1307 Once provisioned, the Client can assert the role it is using as described in 10.4.2, if it has a
1308 certificate role credential.

1309 All provisioned roles are used in ACL enforcement. When a server has multiple roles provisioned
1310 for a client, access to a Resource is granted if it would be granted under any of the roles.

1311 **5.4.5 ACL provisioning**

1312 ACL provisioning shall be performed over a secure connection between the AMS and its Devices.
1313 The AMS maintains an ACL policy for each Device it manages. The AMS shall provision the ACL
1314 policy by updating the Device's ACL Resources.

1315 The AMS shall digitally sign an ACL as part of issuing a /oic/sec/sacl Resource if the Device
1316 supports the /oic/sec/sacl Resource. The public key used by the Device to verify the signature
1317 shall be provisioned by the CMS as needed. A /oic/sec/cred Resource with an asymmetric key
1318 type or signed asymmetric key type is used. The PublicData Property contains the AMS's public
1319 key.

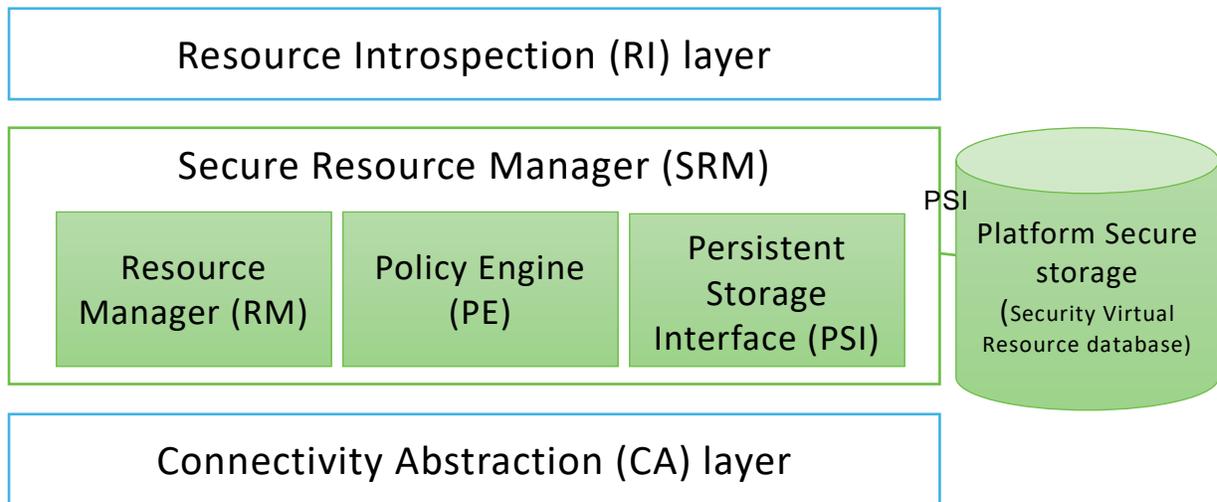
1320 **5.5 Secure Resource Manager (SRM)**

1321 SRM plays a key role in the overall security operation. In short, SRM performs both management
1322 of SVR and access control for requests to access and manipulate Resources. SRM consists of 3
1323 main functional elements:

- 1324 – A Resource manager (RM): responsible for 1) Loading SVRs from persistent storage (using
1325 PSI) as needed. 2) Supplying the Policy Engine (PE) with Resources upon request. 3)
1326 Responding to requests for SVRs. While the SVRs are in SRM memory, the SVRs are in a
1327 format that is consistent with device-specific data store format. However, the RM will use
1328 JSON format to marshal SVR data structures before be passed to PSI for storage, or travel
1329 off-device.

- 1330 – A Policy Engine (PE) that takes requests for access to SVRs and based on access control
- 1331 policies responds to the requests with either "ACCESS_GRANTED" or "ACCESS_DENIED".
- 1332 To make the access decisions, the PE consults the appropriate ACL and looks for best
- 1333 Access Control Entry (ACE) that can serve the request given the subject (Device or role) that
- 1334 was authenticated by DTLS.
- 1335 – Persistent Storage Interface (PSI): PSI provides a set of APIs for the RM to manipulate files in
- 1336 its own memory and storage. The SRM design is modular such that it may be implemented in
- 1337 the Platform's secure execution environment; if available.

1338 Figure 12 depicts OCF's SRM Architecture.



1339

1340 **Figure 12 – OCF's SRM Architecture**

1341 **5.6 Credential Overview**

1342 Devices may use credentials to prove the identity and role(s) of the parties in bidirectional
 1343 communication. Credentials can be symmetric or asymmetric. Each device stores secret and
 1344 public parts of its own credentials where applicable, as well as credentials for other devices that
 1345 have been provided by the DOTS or a CMS. These credentials are then used in the
 1346 establishment of secure communication sessions (e.g. using DTLS) to validate the identities of
 1347 the participating parties. Role credentials are used once an authenticated session is established,
 1348 to assert one or more roles for a device.

1349 Access Tokens are provided to an OCF Cloud once an authenticated session with an OCF Cloud
 1350 is established, to verify the User ID with which the Device is to be associated.

1351

1352 **6 Security for the Discovery Process**

1353 **6.1 Preamble**

1354 The main function of a discovery mechanism is to provide Universal Resource Identifiers (URIs,
1355 called links) for the Resources hosted by the Server, complemented by attributes about those
1356 Resources and possible further link relations. (in accordance to Clause 10 in ISO/IEC 30118-
1357 1:2018)

1358 **6.2 Security Considerations for Discovery**

1359 When defining discovery process, care must be taken that only a minimum set of Resources are
1360 exposed to the discovering entity without violating security of sensitive information or privacy
1361 requirements of the application at hand. This includes both data included in the Resources, as
1362 well as the corresponding metadata.

1363 To achieve extensibility and scalability, this document does not provide a mandate on
1364 discoverability of each individual Resource. Instead, the Server holding the Resource will rely on
1365 ACLs for each Resource to determine if the requester (the Client) is authorized to see/handle any
1366 of the Resources.

1367 The `/oic/sec/acl2` Resource contains ACL entries governing access to the Server hosted
1368 Resources. (See 13.5)

1369 Aside from the privacy and discoverability of Resources from ACL point of view, the discovery
1370 process itself needs to be secured. This document sets the following requirements for the
1371 discovery process:

- 1372 1) Providing integrity protection for discovered Resources.
- 1373 2) Providing confidentiality protection for discovered Resources that are considered sensitive.

1374 The discovery of Resources is done by doing a RETRIEVE operation (either unicast or multicast)
1375 on the known `/oic/res` Resource.

1376 The discovery request is sent over a non-secure channel (multicast or unicast without DTLS), a
1377 Server cannot determine the identity of the requester. In such cases, a Server that wants to
1378 authenticate the Client before responding can list the secure discovery URI (e.g.
1379 `coaps://IP:PORT/oic/res`) in the unsecured `/oic/res` Resource response. This means the secure
1380 discovery URI is by default discoverable by any Client. The Client will then be required to send a
1381 separate unicast request using DTLS to the secure discovery URI.

1382 For secure discovery, any Resource that has an associated ACL2 will be listed in the response to
1383 `/oic/res` Resource if and only if the Client has permissions to perform at least one of the CRUDN
1384 operations (i.e. the bitwise OR of the CRUDN flags must be true).

1385 For example, a Client with Device Id "d1" makes a RETRIEVE request on the `/door` Resource
1386 hosted on a Server with Device Id "d3" where d3 has the ACL2s:

```
1387 {  
1388   "aclist2": [  
1389     {  
1390       "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},  
1391       "resources": [{"href": "/door"}],  
1392       "permission": 2, // RETRIEVE  
1393       "aceid": 1  
1394     }  
1395   ],
```

```

1396     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1397   }
1398   {
1399     "aclist2": [
1400       {
1401         "subject": {"authority": "owner", "role": "owner"}
1402         "resources": [{"href": "/door"}],
1403         "permission": 2, // RETRIEVE
1404         "aceid": 2
1405       }
1406     ],
1407     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1408   }
1409   {
1410     "aclist2": [
1411       {
1412         "subject": {"uuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"},
1413         "resources": [{"href": "/door/lock"}],
1414         "permission": 4, // UPDATE
1415         "aceid": 3
1416       }
1417     ],
1418     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1419   }
1420   {
1421     "aclist2": [
1422       {
1423         "subject": {"conntype": "anon-clear"},
1424         "resources": [{"href": "/light"}],
1425         "permission": 2, // RETRIEVE
1426         "aceid": 4
1427       }
1428     ],
1429     "rowneruuid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1430   }

```

1431 The ACL indicates that Client "d1" has RETRIEVE permissions on the Resource. Hence when
1432 device "d1" does a discovery on the "/oic/res" Resource of the Server "d3", the response will
1433 include the URI of the "/door" Resource metadata. Client "d2" will have access to both the
1434 Resources. ACE2 will prevent "d4" from update.

1435 Discovery results delivered to d1 regarding d3's "/oic/res" Resource from the secure interface:

```

1436 [
1437   {
1438     "href": "/door",
1439     "rt": ["oic.r.door"],
1440     "if": ["oic.if.b", "oic.ll"],

```

```
1441     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1442   }
1443 ]
```

1444 Discovery results delivered to d2 regarding d3's "/oic/res" Resource from the secure interface:

```
1445 [
1446   {
1447     "href": "/door",
1448     "rt": ["oic.r.door"],
1449     "if": ["oic.if.b", "oic.ll"],
1450     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1451   },
1452   {
1453     "href": "/door/lock",
1454     "rt": ["oic.r.lock"],
1455     "if": ["oic.if.b"],
1456     "type": ["application/json", "application/exi+xml"]
1457   }
1458 ]
```

1459 Discovery results delivered to d4 regarding d3's "/oic/res" Resource from the secure interface:

```
1460 [
1461   {
1462     "href": "/door/lock",
1463     "rt": ["oic.r.lock"],
1464     "if": ["oic.if.b"],
1465     "type": ["application/json", "application/exi+xml"],
1466     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
1467   }
1468 ]
```

1469 Discovery results delivered to any device regarding d3's /oic/res Resource from the unsecure interface:

```
1471 [
1472   {
1473     "di": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1",
1474     "href": "/light",
1475     "rt": ["oic.r.light"],
1476     "if": ["oic.if.s"]
1477   }
1478 ]
1479
```

1480 **7 Security Provisioning**

1481 **7.1 Device Identity**

1482 **7.1.1 General Device Identity**

1483 Each Device, which is a logical device, is identified with a Device ID.

1484 Devices shall be identified by a Device ID value that is established as part of device onboarding.
1485 The "/oic/sec/doxm" Resource specifies the Device ID format (e.g. urn:uuid). Device IDs shall be
1486 unique within the scope of operation of the corresponding OCF Security Domain, and should be
1487 universally unique. The DOTS shall ensure Device ID of the new Device is unique within the
1488 scope of the owner's OCF Security Domain. The DOTS shall verify the chosen new device
1489 identifier does not conflict with Device IDs previously introduced into the OCF Security Domain.

1490 Devices maintain an association of Device ID and cryptographic credential using a "/oic/sec/cred"
1491 Resource. Devices regard the "/oic/sec/cred" Resource as authoritative when verifying
1492 authentication credentials of a peer device.

1493 A Device maintains its Device ID in the "/oic/sec/doxm" Resource. It maintains a list of credentials,
1494 both its own and other Device credentials, in the /oic/sec/cred Resource. The device ID can be
1495 used to distinguish between a device's own credential, and credentials for other devices.
1496 Furthermore, the "/oic/sec/cred" Resource may contain multiple credentials for the device.

1497 Device ID shall be:

- 1498 – Unique
- 1499 – Immutable
- 1500 – Verifiable

1501 When using manufacturer certificates, the certificate should bind the ID to the stored secret in the
1502 device as described later in this clause.

1503 A physical Device, referred to as a Platform in OCF documents, may host multiple Devices. The
1504 Platform is identified by a Platform ID. The Platform ID shall be globally unique and inserted in
1505 the device in an integrity protected manner (e.g. inside secure storage or signed and verified).

1506 An OCF Platform may have a secure execution environment, which shall be used to secure
1507 unique identifiers and secrets. If a Platform hosts multiple devices, some mechanism is needed to
1508 provide each Device with the appropriate and separate security.

1509 **7.1.2 Device Identity for Devices with UAID**

1510 **7.1.2.1 Unique Authenticable Identifier**

1511 When a manufacturer certificate is used with certificates chaining to an OCF root CA (as
1512 specified in 7.1.2), the manufacturer shall include a Platform ID inside the certificate subject CN
1513 field. In such cases, the device ID may be created according to the Unique Authenticable
1514 Identifier (UAID) scheme defined in this clause.

1515 For identifying and protecting Devices, the Platform Secure Execution Environment (SEE) may
1516 opt to generate new Dynamic Public Key Pair (DPKP) for each Device it is hosting, or it may opt
1517 to simply use the same public key credentials embedded by manufacturer; Embedded Platform
1518 Credential (EPC). In either case, the Platform SEE will use its Random Number Generator (RNG)
1519 to create a device identity called UAID for each Device. The UAID is generated using either EPC
1520 only or the combination of DPKP and EPC if both are available. When both are available, the
1521 Platform shall use both key pairs to generate the UAID as described in this clause.

1522 The Device ID is formed from the device's public keys and associated OCF Cipher Suite. The
1523 Device ID is formed by:

1524 1) Determining the OCF Cipher Suite of the Dynamic Public Key. The Cipher Suite curve must
1525 match the usage of the AlgorithmIdentifier used in SubjectPublicKeyInfo as intended for use
1526 with Device security mechanisms. Use the encoding of the CipherSuite as the 'csid' value in
1527 the following calculations. If the OCF Cipher Suite for Dynamic Public key is different from the
1528 ciphersuite indicated in the Platform certificate (EPC), the OCF Cipher Suite shall be used
1529 below.

1530 2) From EPC extract the value of embedded public key. The value should correspond to the
1531 value of subjectPublicKey defined in SubjectPublicKeyInfo of the certificate. In the following
1532 we refer to this as EPK. If the public key is extracted from a certificate, validate that the
1533 AlgorithmIdentifier matches the expected value for the CipherSuite within the certificate.

1534 3) From DPKP, extract the value of the public key. The value should correspond to the value of
1535 subjectPublicKey defined in SubjectPublicKeyInfo. In the following we refer to this as DPK.

1536 4) Using the hash for the Cipher Suite calculate: $h = \text{hash}(\text{'uaid'} \mid \text{csid} \mid \text{EPK} \mid \text{DPK} \mid$
1537 $\text{<other_info>})$

1538 Other_info could be 1) device type as indicated in "/oic/d" (could be read-only and set by
1539 manufacturer), 2) in case there are two sets of public key pairs (one embedded, and one
1540 dynamically generated), both public keys would be included.

1541 5) Truncate to 160 bits by taking the leftmost 160 bits of h UAID = $h[0:16]$ # leftmost 16 octets

1542 6) Convert the binary UAID to a ASCII string by $\text{USID} = \text{base27encode}(\text{UAID})$

```
1543 def base_N_encode(octets, alphabet):  
1544     long_int = string_to_int( octets )  
1545     text_out = ''  
1546     while long_int > 0:  
1547         long_int, remainder = divmod(long_int, len(alphabet))  
1548         text_out = alphabet[remainder] + text_out  
1549     return text_out
```

```
1551 b27chars = 'ABCDEFGHJKMNPQRTWXYZ2346789'
```

```
1552 def b27encode(octet_string):  
1553     """Encode a octet string using 27 characters. """  
1554     return base_N_encode(octet_string, _b27chars )
```

1555 7) Append the string value of USID to "urn:usid:" to form the final string value of the Device ID
1556 urn:usid:ABXW....

1557 Whenever the public key is encoded the format described in IETF RFC 7250 for
1558 SubjectPublicKeyInfo shall be used.

1559 7.1.2.2 Validation of UAID

1560 To be able to use the newly generated Device ID (UAID) and public key pair (DPKP), the device
1561 Platform shall use the embedded private key (corresponding to manufacturer embedded public
1562 key and certificate) to sign a token vouching for the fact that it (the Platform) has in fact
1563 generated the DPKP and UAID and thus deferring the liability of the use of the DPKP to the new
1564 device owner. This also allows the ecosystem to extend the trust from manufacturer certificate to
1565 a device issued certificate for use in the new DPKP and UAID. The degree of trust is in
1566 dependent of the level of hardening of the device SEE.

```
1567 Dev_Token=Info, Signature(hash(info))  
1568 Signature algorithm=ECDSA (can be same algorithm as that in EPC or that possible for  
1569 DPKP)  
1570 Hash algorithm=SHA256  
1571 Info=UAID| <Platform ID> | UAID_generation_data | validity
```

1572 UAID_generation_data=data passed to the hash algorithm used to generate UAID.
1573 Validity=validity period in days (how long the token will be valid)

1574 **7.2 Device Ownership**

1575 This is an informative clause. Devices are logical entities that are security endpoints that have an
1576 identity that is authenticable using cryptographic credentials. A Device is "un-owned" when it is
1577 first initialized. Establishing device ownership is a process by which the device asserts its
1578 identity to the DOTS and the DOTS provisions an owner identity. This exchange results in the
1579 device changing its ownership state, thereby preventing a different DOTS from asserting
1580 administrative control over the device.

1581 The ownership transfer process starts with the OBT discovering a new device that is "un-owned"
1582 through examination of the "Owned" Property of the "/oic/sec/doxm" Resource of the new device.
1583 At the end of ownership transfer, the following is accomplished:

- 1584 1) The DOTS shall establish a secure session with new device.
- 1585 2) Optionally asserts any of the following:
 - 1586 a) Proximity (using PIN) of the OBT to the Platform.
 - 1587 b) Manufacturer's certificate asserting Platform vendor, model and other Platform specific
1588 attributes.
- 1589 3) Determines the device identifier.
- 1590 4) Determines the device owner.
- 1591 5) Specifies the device owner (e.g. Device ID of the OBT).
- 1592 6) Provisions the device with owner's credentials.
- 1593 7) Sets the "Owned" state of the new device to TRUE.

1594 NOTE A Device which connects to the OCF Cloud still retains the ownership established at onboarding with the
1595 DOTS.

1596 **7.3 Device Ownership Transfer Methods**

1597 **7.3.1 OTM implementation requirements**

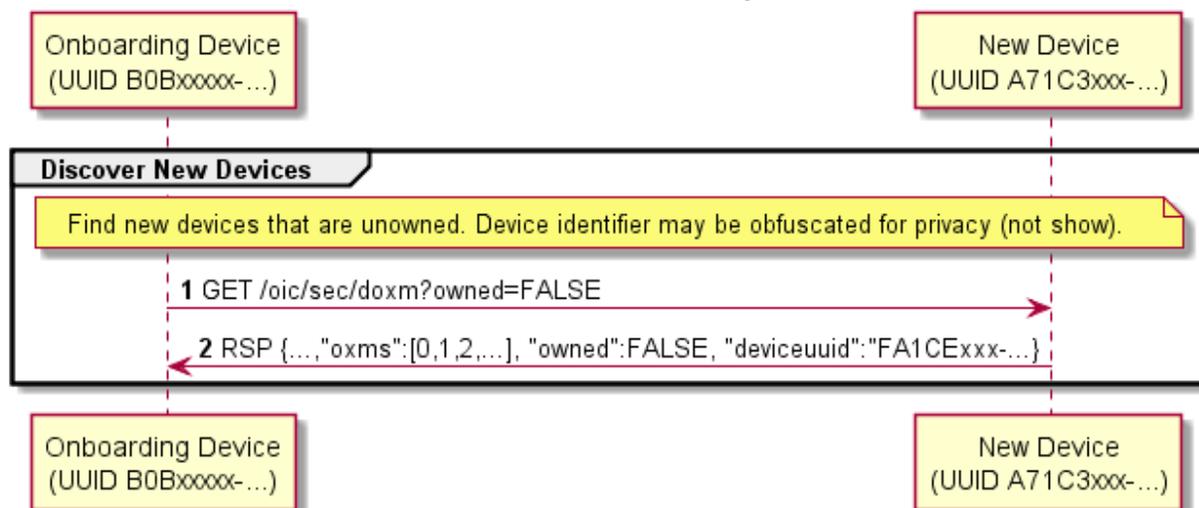
1598 This document provides specifications for several methods for ownership transfer.
1599 Implementation of each individual ownership transfer method is considered optional. However,
1600 each device shall implement at least one of the ownership transfer methods not including vendor
1601 specific methods.

1602 All OTMs included in this document are considered optional. Each vendor is required to choose
1603 and implement at least one of the OTMs specified in this document. The OCF, does however,
1604 anticipate vendor-specific approaches will exist. Should the vendor wish to have interoperability
1605 between a vendor-specific OTM and OBTs from other vendors, the vendor must work directly with
1606 OBT vendors to ensure interoperability. Notwithstanding, standardization of OTMs is the
1607 preferred approach. In such cases, a set of guidelines is provided in 7.3.7 to help vendors in
1608 designing vendor-specific OTMs.

1609 The "/oic/sec/doxm" Resource is extensible to accommodate vendor-defined owner transfer
1610 methods (OTM). The DOTS determines which OC is most appropriate to onboard the new Device.
1611 All OTMs shall represent the onboarding capabilities of the Device using the oxms Property of the
1612 /oic/sec/doxm Resource. The DOTS shall query the Device's supported credential types using the
1613 credtypes Property of the "/oic/sec/cred" Resource. The DOTS and CMS shall provision
1614 credentials according to the credential types supported.

1615 Figure 13 depicts new Device discovery sequence.

Discover New Devices Sequence



1616

1617

Figure 13 – Discover New Device Sequence

1618

1619

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.

1620 Vendor-specific device OTMs shall adhere to the "/oic/sec/doxm" Resource Specification for OCs
 1621 that results from vendor-specific device OTM. Vendor-specific OTM should include provisions for
 1622 establishing trust in the new Device by the OBT an optionally establishing trust in the OBT by the
 1623 new Device.

1624 The new device may have to perform some initialization steps at the beginning of an OTM. For
 1625 example, if the Random PIN Based OTM is initiated, the new device may generate a random PIN
 1626 value. The OBT shall POST to the oxmsel property of /oic/sec/doxm the value corresponding to
 1627 the OTM being used, before performing other OTM steps. This POST notifies the new device that
 1628 ownership transfer is starting.

1629 The end state of a vendor-specific OTM shall allow the new Device to authenticate to the OBT
 1630 and the OBT to authenticate to the new device.

1631 The DOTS may perform additional provisioning steps subsequent to owner transfer success
 1632 leveraging the established OTM session.

1633 After successful OTM, but before placing the newly-onboarded Device in RFNOP, the OBT shall
1634 remove all ACEs where the Subject is "anon-clear" or "auth-crypt", and the Resources array
1635 includes a SVR.

1636 7.3.2 SharedKey Credential Calculation

1637 The SharedKey credential is derived using a PRF that accepts the key_block value resulting from
1638 the DTLS handshake used for onboarding. The new Device and DOTS shall use the following
1639 calculation to ensure interoperability across vendor products:

1640 SharedKey = PRF(Secret, Message);

1641 Where:

- 1642 - PRF shall use TLS 1.2 PRF defined by IETF RFC 5246 clause 5.
- 1643 - Secret is the key_block resulting from the DTLS handshake
 - 1644 ▪ See IETF RFC 5246 Clause 6.3
 - 1645 ▪ The length of key_block depends on cipher suite.
 - 1646 • (e.g. 96 bytes for TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
 - 1647 40 bytes for TLS_PSK_WITH_AES_128_CCM_8)
- 1648 - Message is a concatenation of the following:
 - 1649 ▪ DoxmType string for the current onboarding method (e.g. "oic.sec.doxm.jw")
 - 1650 • See "Clause 13.2.4 for specific DoxmTypes"
 - 1651 ▪ Owner ID is a UUID identifying the device owner identifier and the device that maintains SharedKey.
 - 1652 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
 - 1653 ▪ Device ID is new device's UUID Device ID
 - 1654 • Use raw bytes as specified in IETF RFC 4122 clause 4.1.2
- 1655 - SharedKey Length will be 32 octets.
 - 1656 ▪ If subsequent DTLS sessions use 128 bit encryption cipher suites the leftmost 16 octets will be used.
 - 1657 DTLS sessions using 256 bit encryption cipher suites will use all 32 octets.

1658 7.3.3 Certificate Credential Generation

1659 The Certificate Credential will be used by Devices for secure bidirectional communication. The
1660 certificates will be issued by a CMS or an external certificate authority (CA). This CA will be used
1661 to mutually establish the authenticity of the Device. The onboarding details for certificate
1662 generation will be specified in a later version of this document.

1663 7.3.4 Just-Works OTM

1664 7.3.4.1 Just-Works OTM General

1665 Just-works OTM creates a symmetric key credential that is a pre-shared key used to establish a
1666 secure connection through which a device should be provisioned for use within the owner's OCF
1667 Security Domain. Provisioning additional credentials and Resources is a typical step following
1668 ownership establishment. The pre-shared key is called SharedKey.

1669 The DOTS shall select the Just-works OTM and establish a DTLS session using a ciphersuite
1670 defined for the Just-works OTM.

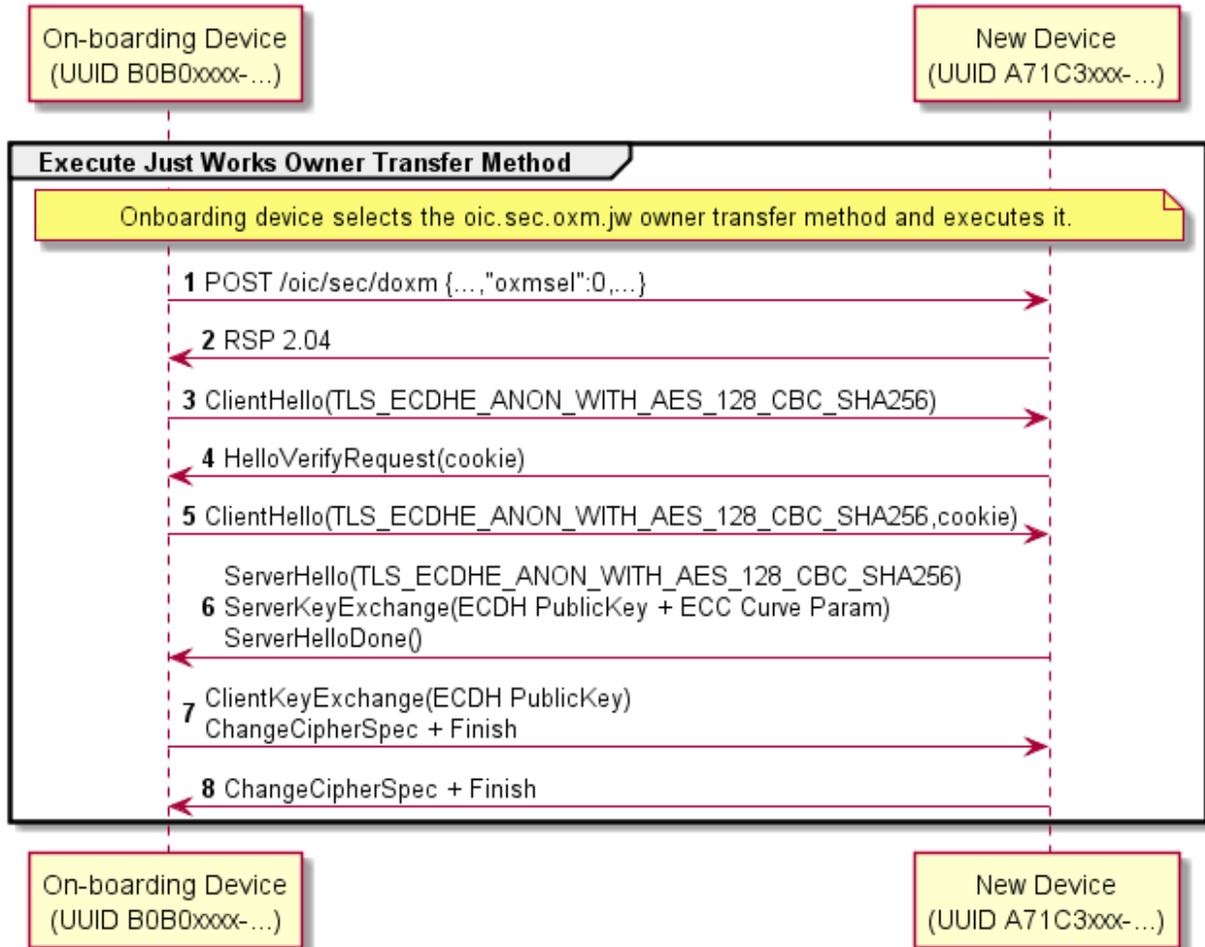
1671 The following OCF-defined vendor-specific ciphersuites are used for the Just-works OTM.

1672 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,
1673 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

1674 These are not registered in IANA, the ciphersuite values are assigned from the reserved area for
1675 private use (0xFF00 ~ 0xFFFF). The assigned values are 0xFF00 and 0xFF01, respectively.

1676 Just Works OTM sequence is shown in Figure 14 and steps described in Table 2.

Perform Just-Works Owner Transfer Method



1677

1678

1679

Figure 14 – A Just Works OTM

Table 2 – A Just Works OTM Details

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

1680

7.3.4.2 Security Considerations

1681 Anonymous Diffie-Hellman key agreement is subject to a man-in-the-middle attacker. Use of this
 1682 method presumes that both the OBT and the new device perform the "just-works" method
 1683 assumes onboarding happens in a relatively safe environment absent of an attack device.

1684 This method doesn't have a trustworthy way to prove the device ID asserted is reliably bound to
 1685 the device.

1686 The new device should use a temporal device ID prior to transitioning to an owned device while it
1687 is considered a guest device to prevent privacy sensitive tracking. The device asserts a non-
1688 temporal device ID that could differ from the temporal value during the secure session in which
1689 owner transfer exchange takes place. The OBT will verify the asserted Device ID does not
1690 conflict with a Device ID already in use. If it is already in use the existing credentials are used to
1691 establish a secure session.

1692 An un-owned Device that also has established device credentials might be an indication of a
1693 corrupted or compromised device.

1694 **7.3.5 Random PIN Based OTM**

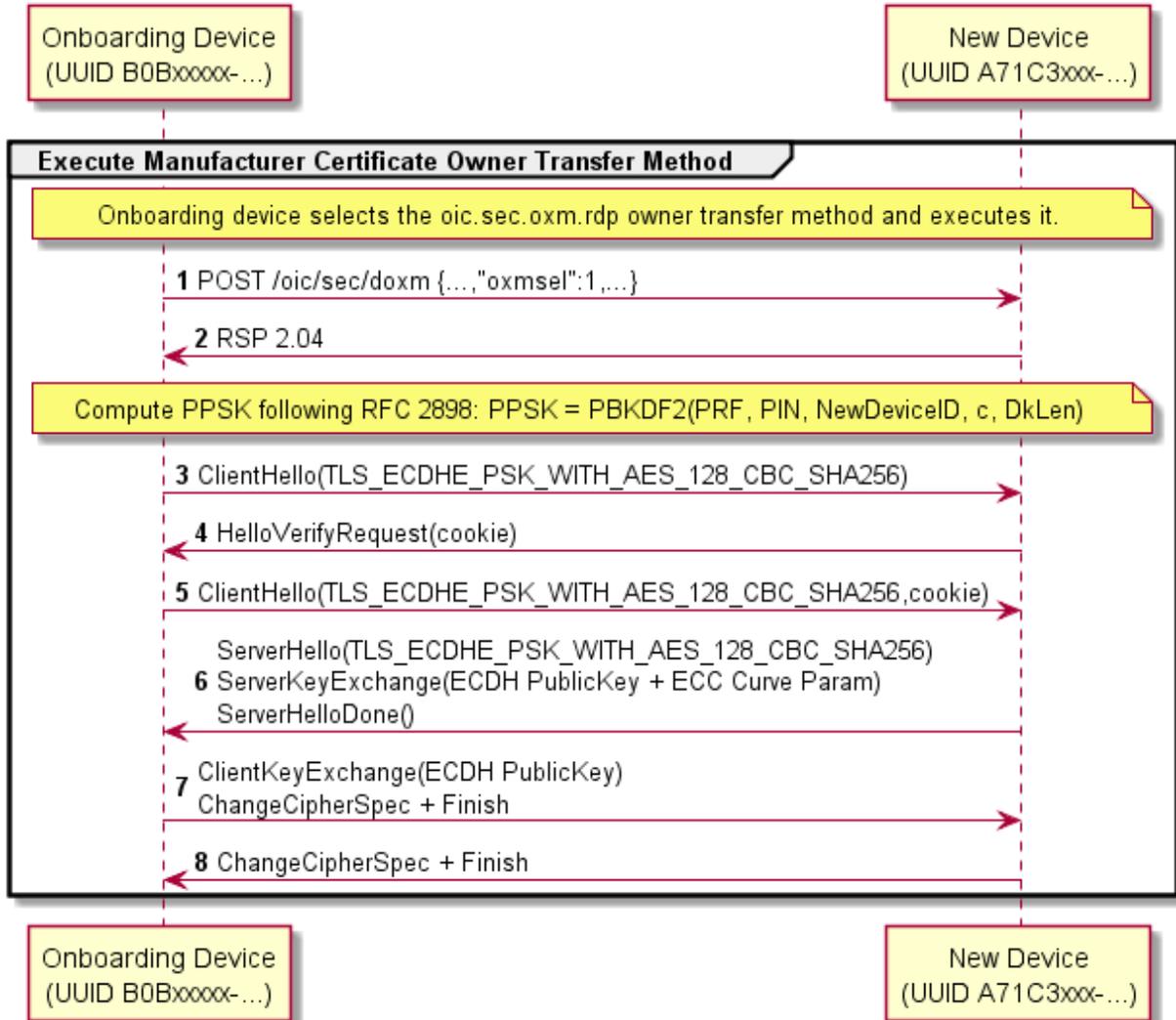
1695 **7.3.5.1 Random PIN OTM General**

1696 The Random PIN method establishes physical proximity between the new device and the OBT
1697 can prevent man-in-the-middle attacks. The Device generates a random number that is
1698 communicated to the OBT over an out-of-band channel. The definition of out-of-band
1699 communications channel is outside the scope of the definition of device OTMs. The OBT and new
1700 Device use the PIN in a key exchange as evidence that someone authorized the transfer of
1701 ownership by having physical access to the new Device via the out-of-band-channel.

1702 **7.3.5.2 Random PIN Owner Transfer Sequence**

1703 Random PIN-based OTM sequence is shown in Figure 15 and steps described in Table 3.

Perform Random PIN Device Owner Transfer Method



1704

1705

1706

1707

Figure 15 – Random PIN-based OTM

Table 3 – Random PIN-based OTM Details

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

1708 The random PIN-based device OTM uses a pseudo-random function (PBKDF2) defined by IETF
1709 RFC 2898 and a PIN exchanged via an out-of-band method to generate a pre-shared key. The
1710 PIN-authenticated pre-shared key (PPSK) is supplied to TLS ciphersuites that accept a PSK.

1711 PPSK = PBKDF2(PRF, PIN, Device ID, c, dkLen)

1712 The PBKDF2 function has the following parameters:

1713 - PRF – Uses the TLS 1.2 PRF defined by IETF RFC 5246.

1714 - PIN – obtain via out-of-band channel.

1715 - Device ID – UUID of the new device.

1716 Use raw bytes as specified in IETF RFC 4122 clause 4.1.2

1717 - c – Iteration count initialized to 1000

1718 - dkLen – Desired length of the derived PSK in octets.

1719 7.3.5.3 Security Considerations

1720 Security of the Random PIN mechanism depends on the entropy of the PIN. Using a PIN with
1721 insufficient entropy may allow a man-in-the-middle attack to recover any long-term credentials
1722 provisioned as a part of onboarding. In particular, learning provisioned symmetric key credentials,
1723 allows an attacker to masquerade as the onboarded device.

1724 It is recommended that the entropy of the PIN be enough to withstand an online brute-force
1725 attack, 40 bits or more. For example, a 12-digit numeric PIN, or an 8-character alphanumeric (0-
1726 9a-z), or a 7 character case-sensitive alphanumeric PIN (0-9a-zA-Z). A man-in-the-middle attack
1727 (MITM) is when the attacker is active on the network and can intercept and modify messages
1728 between the OBT and device. In the MITM attack, the attacker must recover the PIN from the key
1729 exchange messages in "real time", i.e., before the peers time out and abort the connection
1730 attempt. Having recovered the PIN, he can complete the authentication step of key exchange.
1731 The guidance given here calls for a minimum of 40 bits of entropy, however, the assurance this
1732 provides depends on the resources available to the attacker. Given the parallelizable nature of a
1733 brute force guessing attack, the attack enjoys a linear speedup as more cores/threads are added.
1734 A more conservative amount of entropy would be 64 bits. Since the Random PIN OTM requires
1735 using a DTLS ciphersuite that includes an ECDHE key exchange, the security of the Random PIN
1736 OTM is always at least equivalent to the security of the JustWorks OTM.

1737 The Random PIN OTM also has an option to use PBKDF2 to derive key material from the PIN.
1738 The rationale is to increase the cost of a brute force attack, by increasing the cost of each guess
1739 in the attack by a tuneable amount (the number of PBKDF2 iterations). In theory, this is an
1740 effective way to reduce the entropy requirement of the PIN. Unfortunately, it is difficult to quantify
1741 the reduction, since an X-fold increase in time spent by the honest peers does not directly
1742 translate to an X-fold increase in time by the attacker. This asymmetry is because the attacker
1743 may use specialized implementations and hardware not available to honest peers. For this
1744 reason, when deciding how much entropy to use for a PIN, it is recommended that implementers
1745 assume PBKDF2 provides no security, and ensure the PIN has sufficient entropy.

1746 The Random PIN device OTM security depends on an assumption that a secure out-of-band
1747 method for communicating a randomly generated PIN from the new device to the OBT exists. If
1748 the OOB channel leaks some or the entire PIN to an attacker, this reduces the entropy of the PIN,
1749 and the attacks described above apply. The out-of-band mechanism should be chosen such that
1750 it requires proximity between the OBT and the new device. The attacker is assumed to not have
1751 compromised the out-of-band-channel. As an example OOB channel, the device may display a
1752 PIN to be entered into the OBT software. Another example is for the device to encode the PIN as
1753 a 2D barcode and display it for a camera on the OBT device to capture and decode.

1754 **7.3.6 Manufacturer Certificate Based OTM**

1755 **7.3.6.1 Manufacturer Certificate Based OTM General**

1756 The manufacturer certificate-based OTM shall use a certificate embedded into the device by the
1757 manufacturer and may use a signed OBT, which determines the Trust Anchor between the device
1758 and the OBT.

1759 Manufacturer embedded certificates do not necessarily need to chain to an OCF Root CA trust
1760 anchor.

1761 For some environments, policies or administrators, additional information about device
1762 characteristics may be sought. This list of additional attestations that OCF may or may not have
1763 tested (understanding that some attestations are incapable of testing or for which testing may be
1764 infeasible or economically unviable) can be found under the OCF Security Claims x509.v3
1765 extension described in 9.4.2.2.6.

1766 When utilizing certificate-based ownership transfer, devices shall utilize asymmetric keys with
1767 certificate data to authenticate their identities with the OBT in the process of bringing a new
1768 device into operation on an OCF Security Domain. The onboarding process involves several
1769 discrete steps:

1770 1) Pre-on-board conditions

1771 a) The credential element of the Device's credential Resource (/oic/sec/cred) containing the
1772 manufacturer certificate shall be identified by the properties:

1773 i) the subject Property shall refer to the Device

1774 ii) the credusage Property shall contain the string "oic.sec.cred.mfgcert" to indicate that
1775 the credential contains a manufacturer certificate

1776 b) The manufacturer certificate chain shall be contained in the identified credential element's
1777 publicdata Property.

1778 c) The device shall contain a unique and immutable ECC asymmetric key pair.

1779 d) If the device requires authentication of the OBT as part of ownership transfer, it is
1780 presumed that the OBT has been registered and has obtained a certificate for its unique
1781 and immutable ECC asymmetric key pair signed by the predetermined Trust Anchor.

1782 e) User has configured the OBT app with network access info and account info (if any).

1783 2) The OBT shall authenticate the Device using ECDSA to verify the signature. Additionally, the
1784 Device may authenticate the OBT to verify the OBT signature.

1785 3) If authentication fails, the Device shall indicate the reason for failure and return to the Ready
1786 for OTM state. If authentication succeeds, the device and OBT shall establish an encrypted
1787 link in accordance with the negotiated cipher suite.

1788 **7.3.6.2 Certificate Profiles**

1789 See 9.4.2 for details.

1790 **7.3.6.3 Certificate Owner Transfer Sequence Security Considerations**

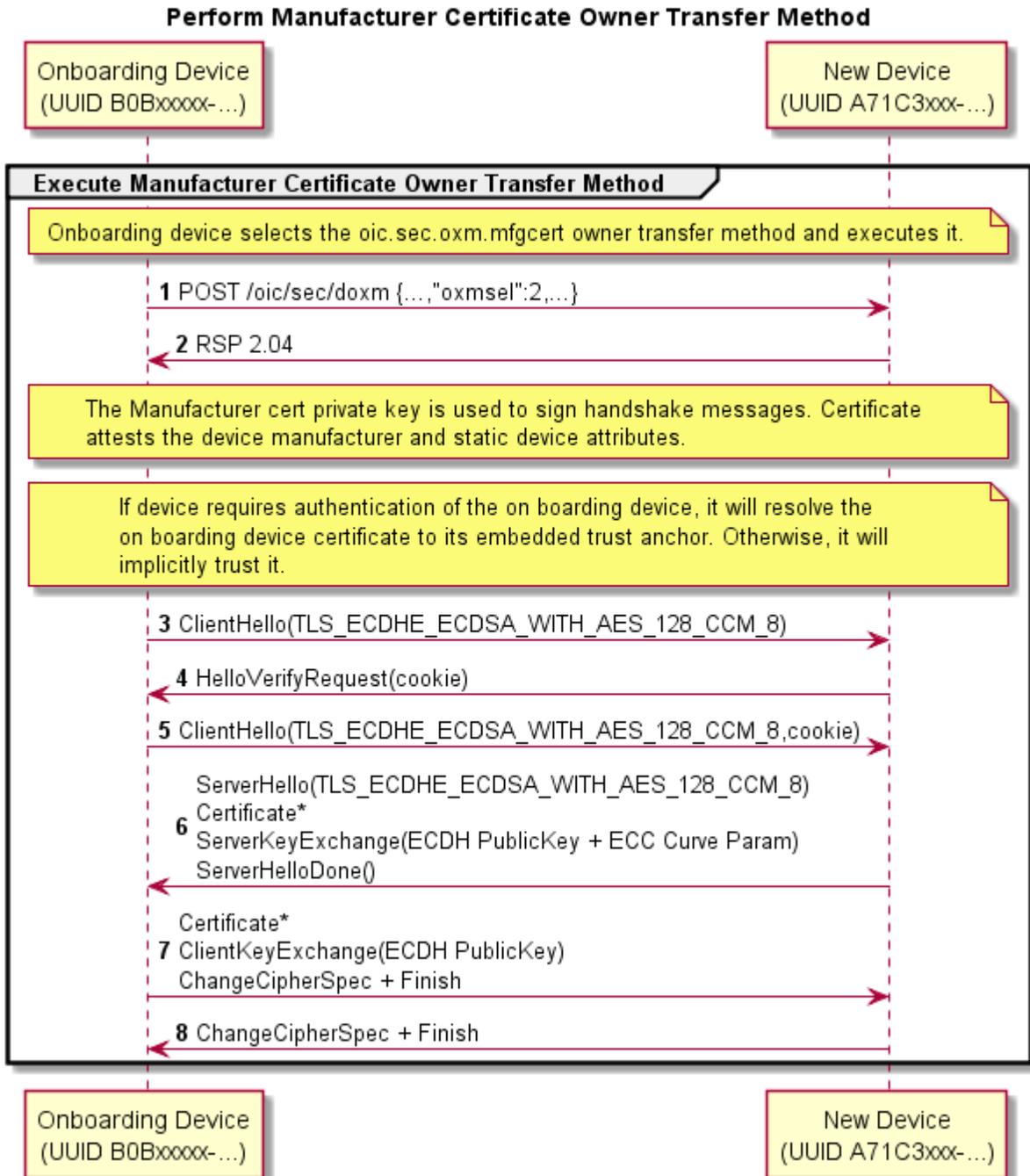
1791 In order for full, mutual authentication to occur between the device and the OBT, both the device
1792 and OBT must be able to trace back to a mutual Trust Anchor or Certificate Authority. This
1793 implies that OCF may need to obtain services from a Certificate Authority (e.g. Symantec,
1794 Verisign, etc.) to provide ultimate Trust Anchors from which all subsequent OCF Trust Anchors
1795 are derived.

1796 The OBT shall authenticate the device during onboarding. However, the device is not required to
1797 authenticate the OBT due to potential resource constraints on the device.

1798 In the case where the Device does NOT authenticate the OBT software, there is the possibility of
 1799 malicious OBT software unwittingly deployed by users, or maliciously deployed by an adversary,
 1800 which can compromise OCF Security Domain access credentials and/or personal information.

1801 **7.3.6.4 Manufacturer Certificate Based OTM Sequence**

1802 Random PIN-based OTM sequence is shown in Figure 16 and steps described in Table 4.



1803

1804

Figure 16 – Manufacturer Certificate Based OTM Sequence

1805

1806

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.

1807 **7.3.6.5 Security Considerations**

1808 The manufacturer certificate private key is embedded in the Platform with a sufficient degree of
1809 assurance that the private key cannot be compromised.

1810 The Platform manufacturer issues the manufacturer certificate and attests the private key
1811 protection mechanism.

1812 **7.3.7 Vendor Specific OTMs**

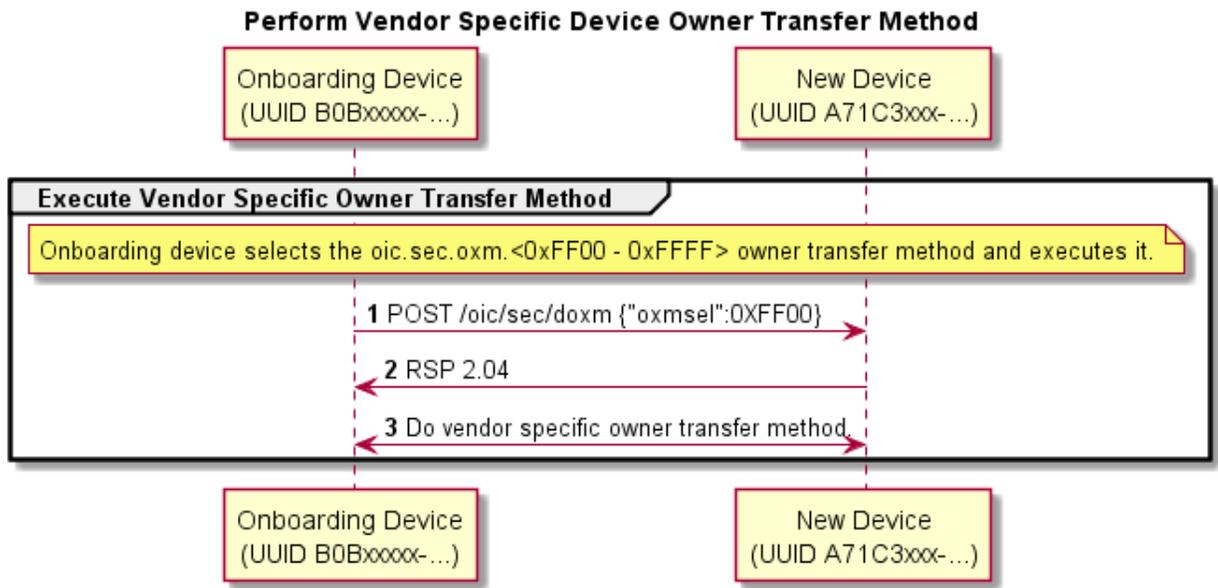
1813 **7.3.7.1 Vendor Specific OTM General**

1814 The OCF anticipates situations where a vendor will need to implement an OTM that
1815 accommodates manufacturing or Device constraints. The Device OTM resource is extensible for
1816 this purpose. Vendor-specific OTMs must adhere to a set of conventions that all OTMs follow.

- 1817 – The OBT must determine which credential types are supported by the Device. This is
1818 accomplished by querying the Device's /oic/sec/doxm Resource to identify supported
1819 credential types.
- 1820 – The OBT provisions the Device with OC(s).
- 1821 – The OBT supplies the Device ID and credentials for subsequent access to the OBT.
- 1822 – The OBT will supply second carrier settings sufficient for accessing the owner's OCF Security
1823 Domain subsequent to ownership establishment.
- 1824 – The OBT may perform additional provisioning steps but must not invalidate provisioning tasks
1825 to be performed by a security service.

1826 **7.3.7.2 Vendor-specific Owner Transfer Sequence Example**

1827 Random PIN-based OTM sequence example is shown in Figure 17 and steps described in
1828 Table 5.



1829

1830

Figure 17 – Vendor-specific Owner Transfer Sequence

1831

1832

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

1833

7.3.7.3 Security Considerations

1834

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

1835

7.3.8 Establishing Owner Credentials

1836

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

1837

1838

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

1839

1840

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

1841

1842

1843

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

1844

1) The OBT shall establish the Device ID and Device owner uuid - Figure 18 and Table 6

1845

2) The OBT then establishes Device's OC - Figure 19 and Table 7. This can be either:

1846

a) Symmetric credential - Figure 20 and Table 8.

1847

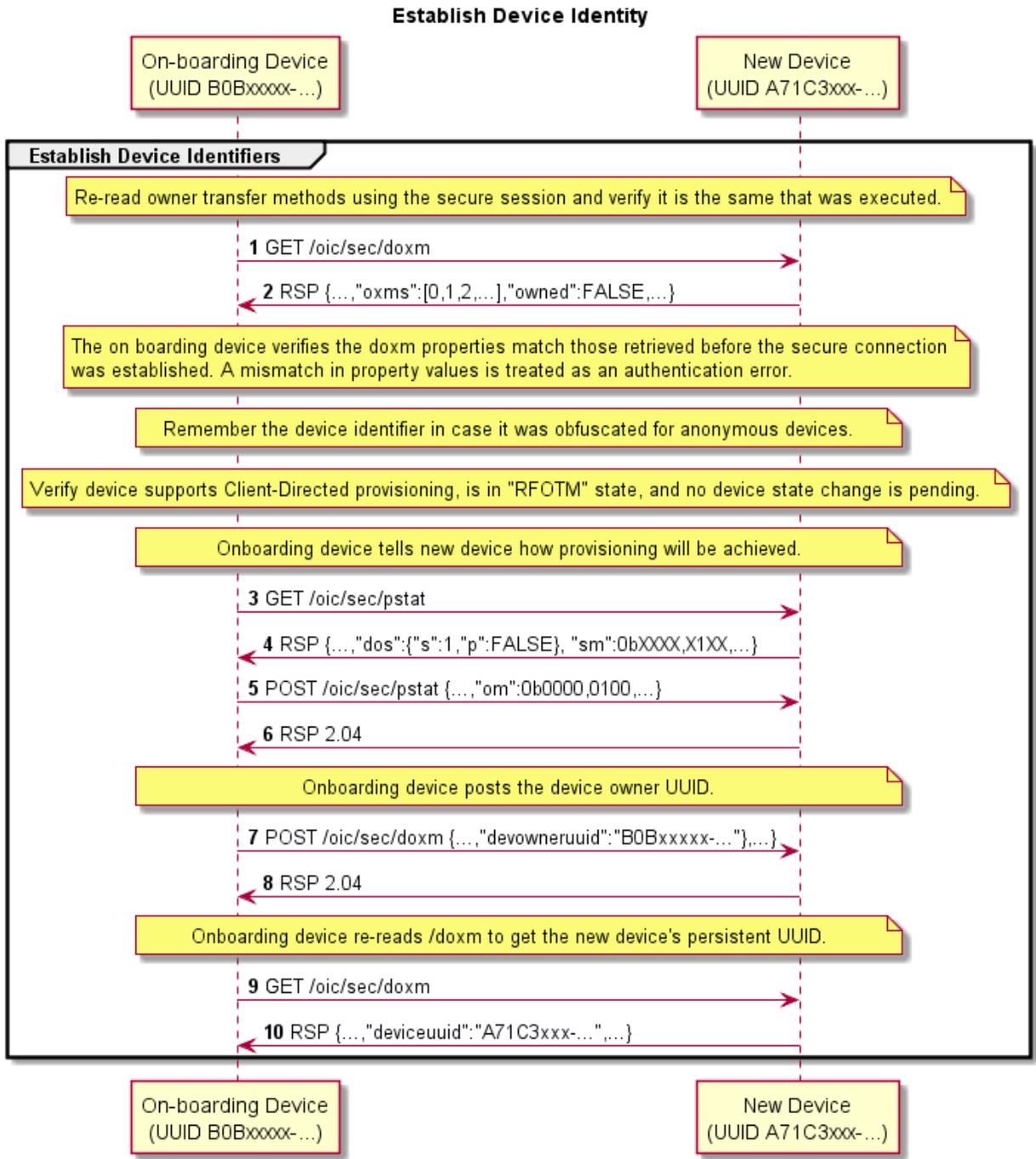
b) Asymmetric credential - Figure 21 and Table 9.

1848

3) Configure Device services - Figure 22 and Table 10.

1849

4) Configure Device for peer to peer interaction - Figure 23 and Table 11.



1851

1852

1853

1854

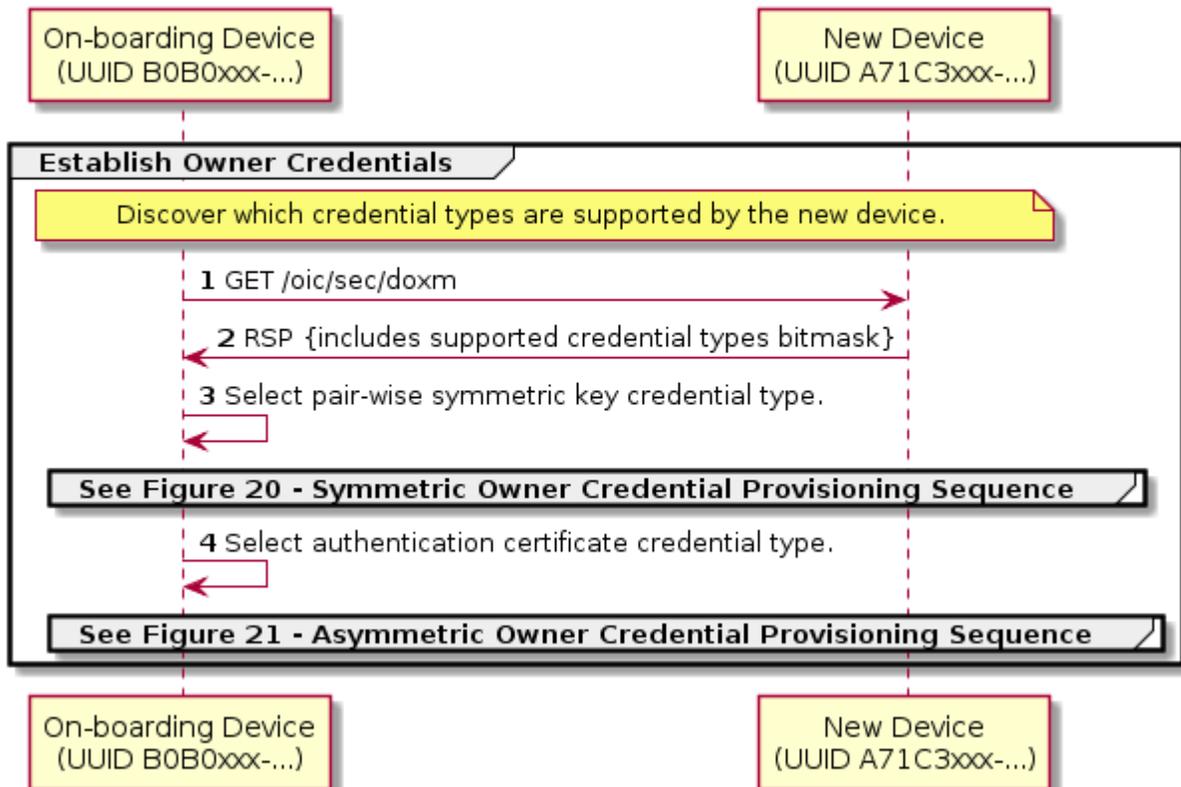
Figure 18 – Establish Device Identity Flow

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



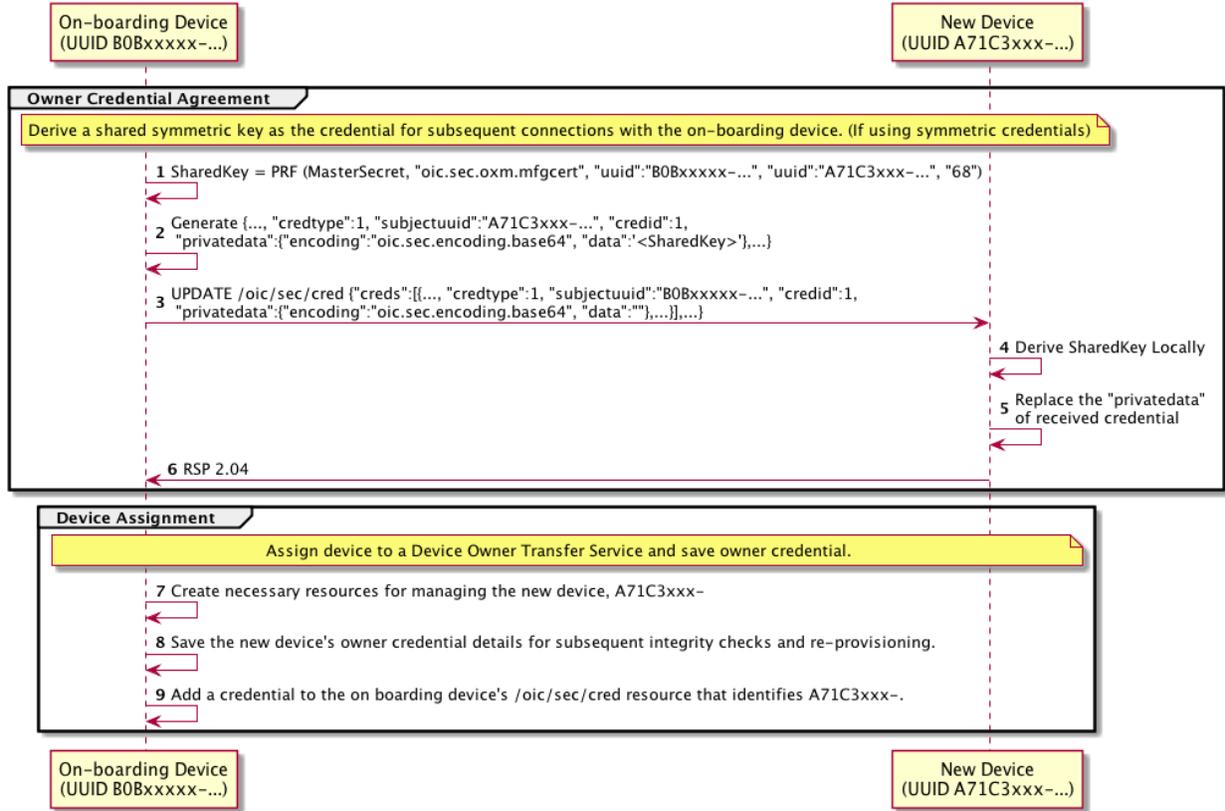
1855
1856
1857
1858

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



1859

1860

1861

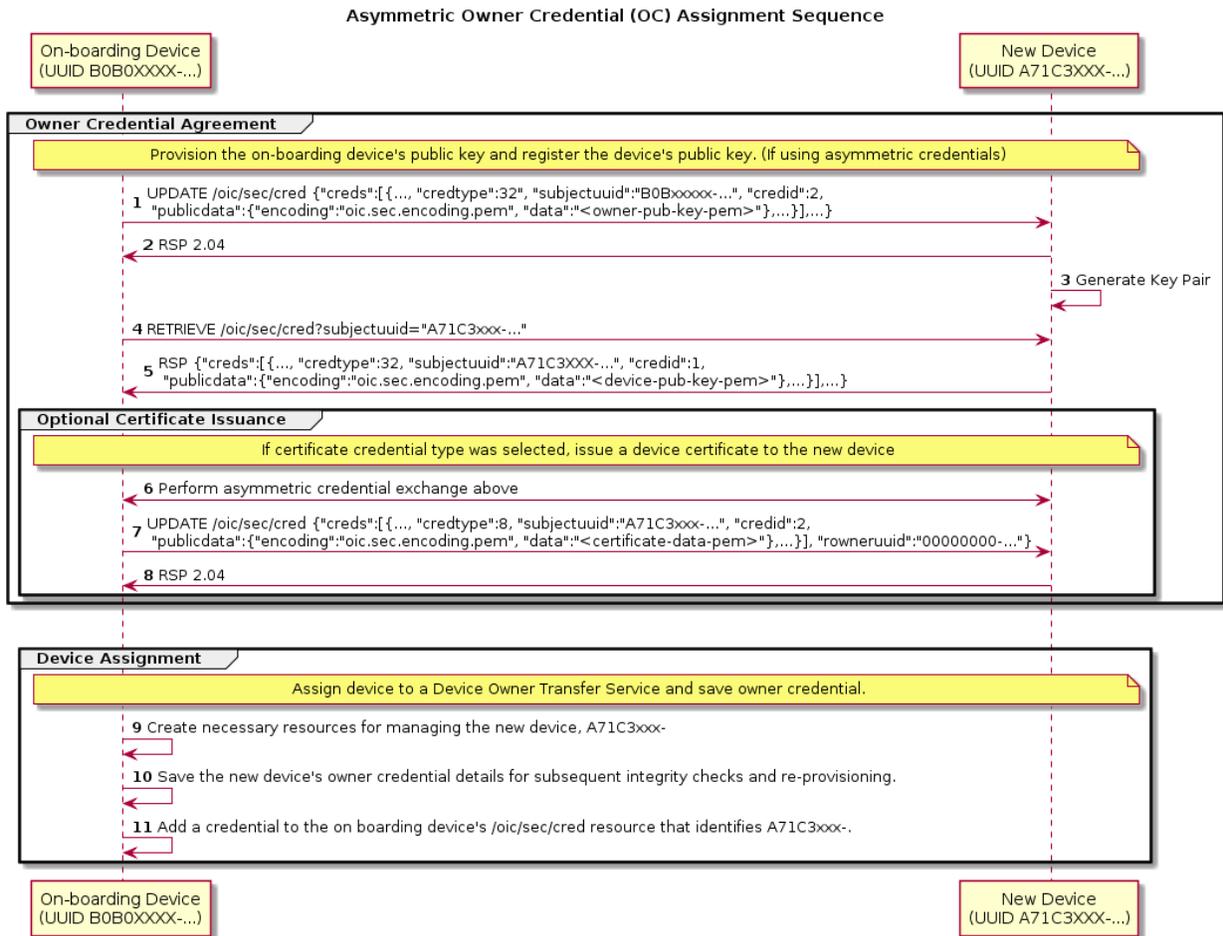
1862

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.

- 1863 In particular, if the OBT selects symmetric owner credentials:
- 1864 – The OBT shall generate a Shared Key using the SharedKey Credential Calculation method
 - 1865 described in 7.3.2.
 - 1866 – The OBT shall send an empty key to the new Device's /oic/sec/cred Resource, identified as a
 - 1867 symmetric pair-wise key.
 - 1868 – Upon receipt of the OBT's symmetric owner credential, the new Device shall independently
 - 1869 generate the Shared Key using the SharedKey Credential Calculation method described in
 - 1870 7.3.2 and store it with the owner credential.
 - 1871 – The new Device shall use the Shared Key owner credential(s) stored via the /oic/sec/cred
 - 1872 Resource to authenticate the owner during subsequent connections.



1873

1874

Figure 21 – Asymmetric Owner Credential Provisioning Sequence

1875

1876

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

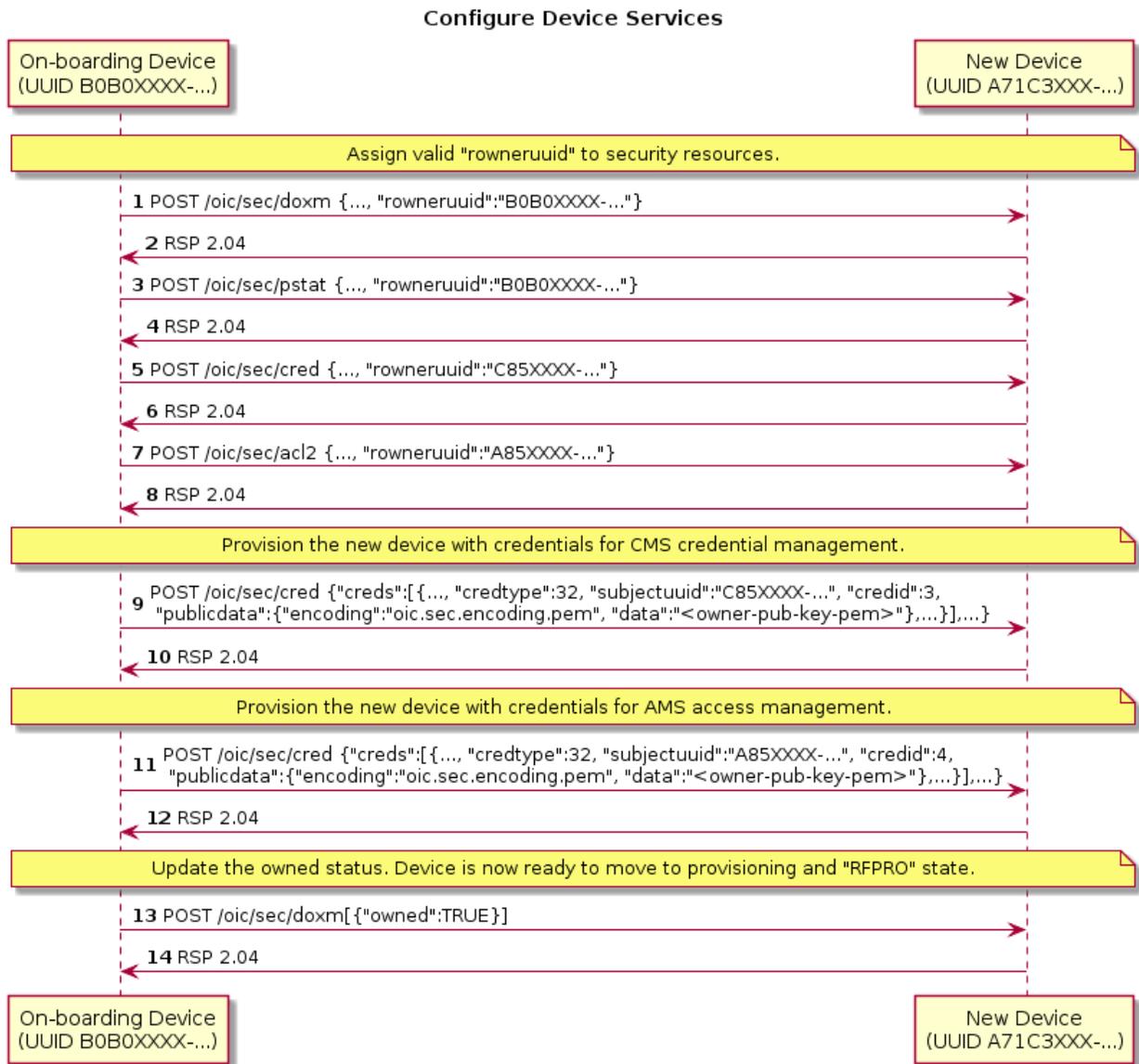
	authenticate the OBT. The new device creates a credential Resource Property set based on the public key generated.
3	The new Device creates an asymmetric key pair.
4, 5	The OBT reads the new Device's asymmetric type credential Resource Property set generated at step 25. It may be used subsequently to 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.

1877 If the OBT selects asymmetric owner credentials:

- 1878 – The OBT shall add its public key to the new Device's /oic/sec/cred Resource, identified as an
1879 Asymmetric Encryption Key.
- 1880 – The OBT shall query the /oic/sec/cred Resource from the new Device, supplying the new
1881 Device's UUID via the SubjectID query parameter. In response, the new Device shall return
1882 the public Asymmetric Encryption Key, which the OBT shall retain for future owner
1883 authentication of the new Device.

1884 If the OBT selects certificate owner credentials:

- 1885 – The OBT shall create a certificate or certificate chain with the leaf certificate containing the
1886 public key returned by the new Device, signed by a mutually-trusted CA, and complying with
1887 the Certificate Credential Generation requirements defined in 7.3.3.
- 1888 – The OBT shall add the newly-created certificate chain to the /oic/sec/cred Resource,
1889 identified as an Asymmetric Signing Key with Certificate.



1890

1891

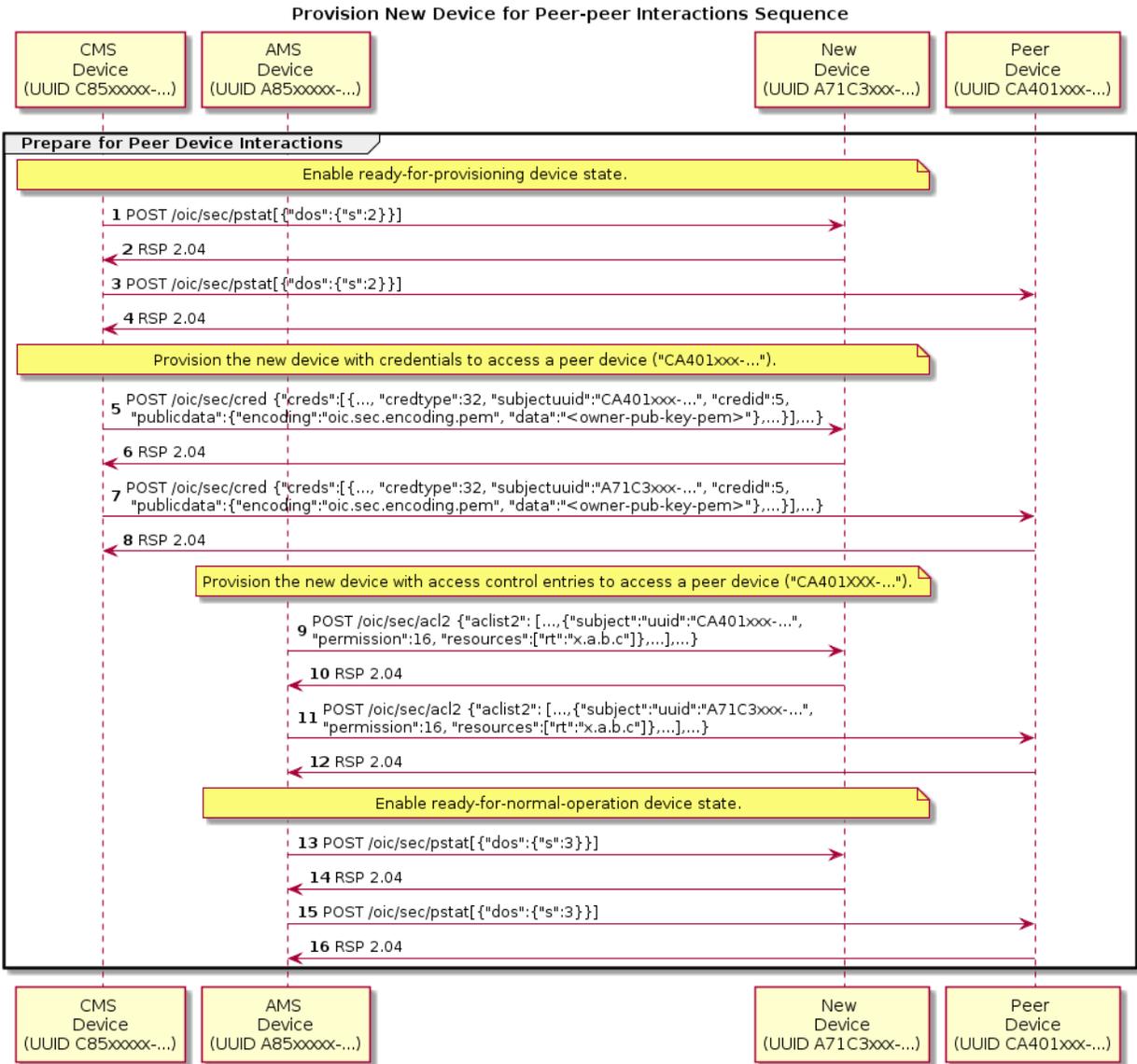
1892

1893

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.



1894

1895

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

1896

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.

1897 **7.3.9 Security considerations regarding selecting an Ownership Transfer Method**

1898 An OBT and/or OBT's operator might have strict requirements for the list of OTMs that are
1899 acceptable when transferring ownership of a new Device. Some of the factors to be considered
1900 when determining those requirements are:

- 1901 – The security considerations described for each of the OTMs
- 1902 – The probability that a man-in-the-middle attacker might be present in the environment used to
1903 perform the Ownership Transfer

1904 For example, the operator of an OBT might require that all of the Devices being onboarded
1905 support either the Random PIN or the Manufacturer Certificate OTM.

1906 When such a local OTM policy exists, the OBT should try to use just the OTMs that are
1907 acceptable according to that policy, regardless of the doxm contents obtained during step 1 from
1908 the sequence diagram above (GET "/oic/sec/doxm"). If step 1 is performed over an
1909 unauthenticated and/or unencrypted connection between the OBT and the Device, the contents of
1910 the response to the GET request might have been tampered by a man-in-the-middle attacker. For
1911 example, the list of OTMs supported by the new Device might have been altered by the attacker.

1912 Also, a man-in-the-middle attacker can force the DTLS session between the OBT and the new
1913 Device to fail. In such cases, the OBT has no way of determining if the session failed because
1914 the new Device doesn't support the OTM selected by the OBT, or because a man-in-the-middle
1915 injected such a failure into the communication between the OBT and the new Device.

1916 The current version of this document leaves the design and user experience related to the OTM
1917 policy as OBT implementation details.

1918 **7.3.10 Security Profile Assignment**

1919 OCF Devices may have been evaluated according to an OCF Security Profile. Evaluation results
1920 could be accessed from a manufacturer's certificate, OCF web server or other public repository.
1921 The DOTS reviews evaluation results to determine which OCF Security Profiles the OCF Device
1922 is authorized to possess and configures the Device with the subset of evaluated security profiles
1923 best suited for the OCF Security Domain owner's intended segmentation strategy.

1924 The OCF Device vendor shall set a manufacturer default value for the "supportedprofiles"
1925 Property of the "/oic/sec/sp" Resource to match those approved by OCF's testing and certification
1926 process. The "currentprofile" Property of the "/oic/sec/sp" Resource shall be set to one of the
1927 values contained in the "supportedprofiles". The manufacturer default value shall be re-asserted
1928 when the Device transitions to RESET Device State.

1929 The OCF Device shall only allow the /oic/sec/sp Resource to be updated when the Device is in
1930 one of the following Device States: RFOTM, RFPRO, SRESET and may not allow any update as
1931 directed by a Security Profile.

1932 The DOTS may update the "supportedprofiles" Property of the "/oic/sec/sp" Resource with a
1933 subset of the OCF Security Profiles values the Device achieved as part of OCF Conformance
1934 testing. The DOTS may locate conformance results by inspecting manufacturer certificates
1935 supplied with the OCF Device by selecting the "credusage" Property of the /oic/sec/cred
1936 Resource having the value of "oic.sec.cred.mfgcert". The DOTS may further locate conformance
1937 results by visiting a well-known OCF web site URI corresponding to the ocCPLAttributes
1938 extension fields (clause 9.4.2.2.7). The DOTS may select a subset of Security Profiles (from
1939 those evaluated by OCF conformance testing) based on a local policy.

1940 As part of onboarding (while the OTM session is active) the DOTS should configure ACE entries
1941 to allow DOTS access subsequent to onboarding.

1942 The DOTS should update the "currentprofile" Property of the "/oic/sec/sp" Resource with the
1943 value that most correctly depicts the OCF Security Domain owner's intended Device deployment
1944 strategy.

1945 The CMS may issue role credentials using the Security Profile value (e.g. the "sp-blue-v0 OID")
1946 to indicate the OCF Security Domain owner's intention to segment the OCF Security Domain
1947 according to a Security Profile. The CMS retrieves the supportedprofiles Property of the
1948 "/oic/sec/sp" Resource to select role names corroborated with the Device's supported Security
1949 Profiles when issuing role credentials.

1950 If the CMS issues role credentials based on a Security Profile, the AMS supplies access control
1951 entries that include the role designation(s).

1952 **7.4 Provisioning**

1953 **7.4.1 Provisioning Flows**

1954 **7.4.1.1 Provisioning Flows General**

1955 As part of onboarding a new Device a secure channel is formed between the new Device and the
1956 OBT. Subsequent to the Device ownership status being changed to "owned", there is an
1957 opportunity to begin provisioning. The OBT decides how the new Device will be managed going
1958 forward and provisions the support services that should be subsequently used to complete
1959 Device provisioning and on-going Device management.

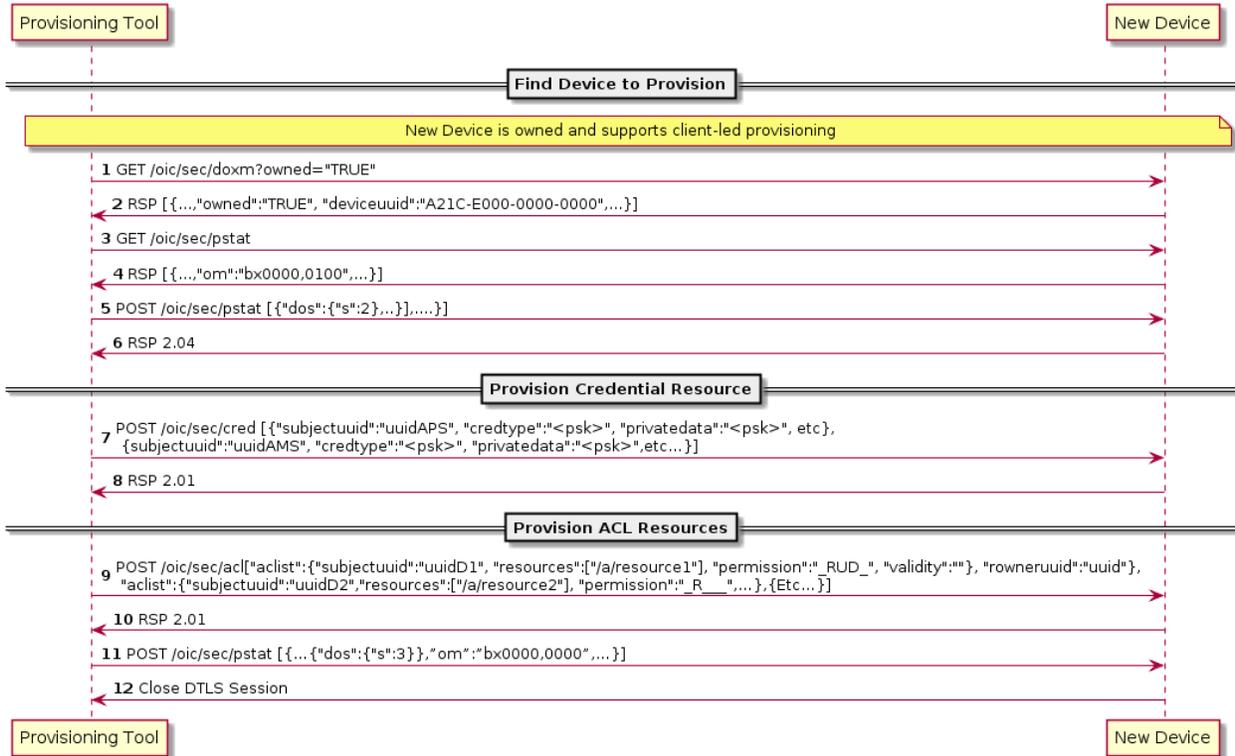
1960 The Device employs a Server-directed or Client-directed provisioning strategy. The /oic/sec/pstat
1961 Resource identifies the provisioning strategy and current provisioning status. The provisioning
1962 service should determine which provisioning strategy is most appropriate for the OCF Security
1963 Domain. See 13.8 for additional detail.

1964 **7.4.1.2 Client-directed Provisioning**

1965 Client-directed provisioning relies on a provisioning service that identifies Servers in need of
1966 provisioning then performs all necessary provisioning duties.

1967 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



1968
1969
1970

Figure 24 – Example of Client-directed provisioning

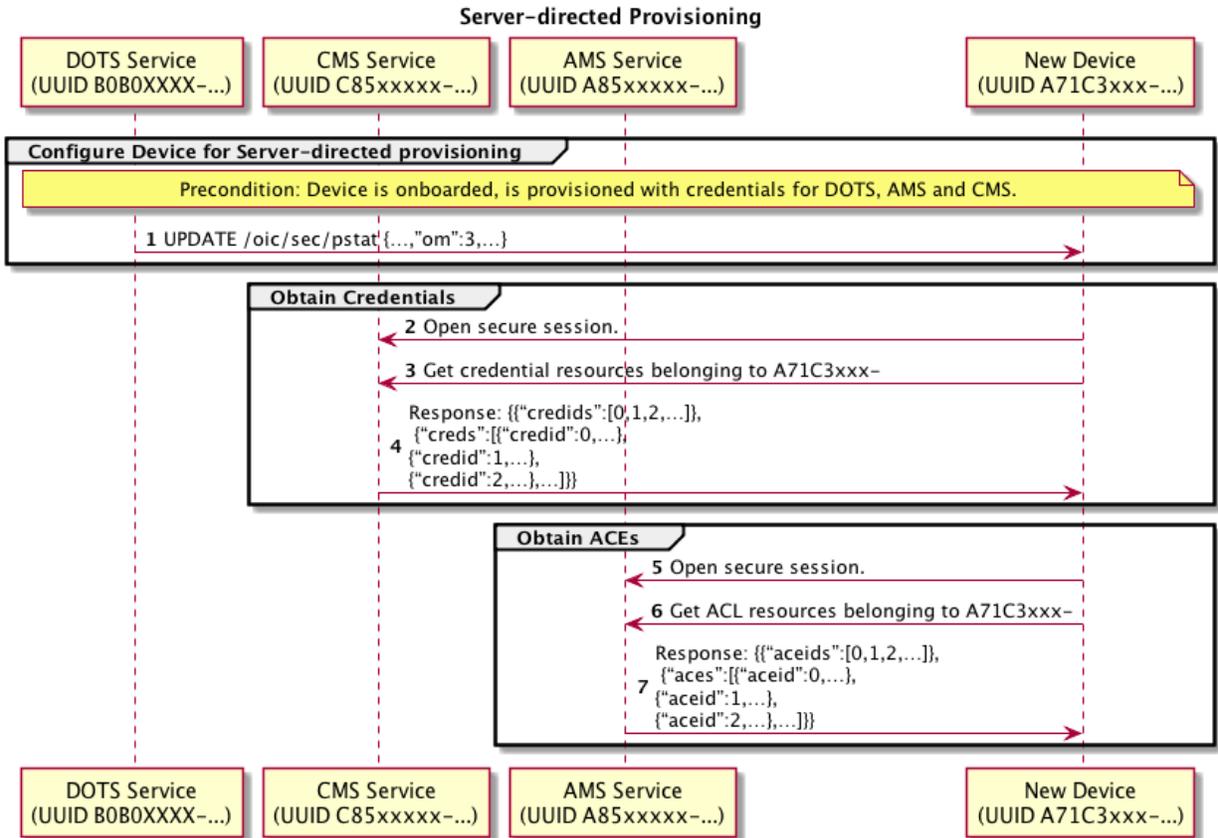
Table 12 – Steps describing Client -directed provisioning

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.

1971 **7.4.1.3 Server-directed Provisioning**

1972 Server-directed provisioning relies on the Server (i.e. New Device) for directing much of the
 1973 provisioning work. As part of the onboarding process the support services used by the Server to
 1974 seek additional provisioning are provisioned. The New Device uses a self-directed, state-driven
 1975 approach to analyze current provisioning state, and tries to drive toward target state. This
 1976 example assumes a single support service is used to provision the new Device.

1977 An example of Client-directed provisioning is shown in Figure 25 and steps described in Table 13.



1978
 1979 **Figure 25 – Example of Server-directed provisioning using a single provisioning service**

1980 **Table 13 – Steps for Server-directed provisioning using a single provisioning service**

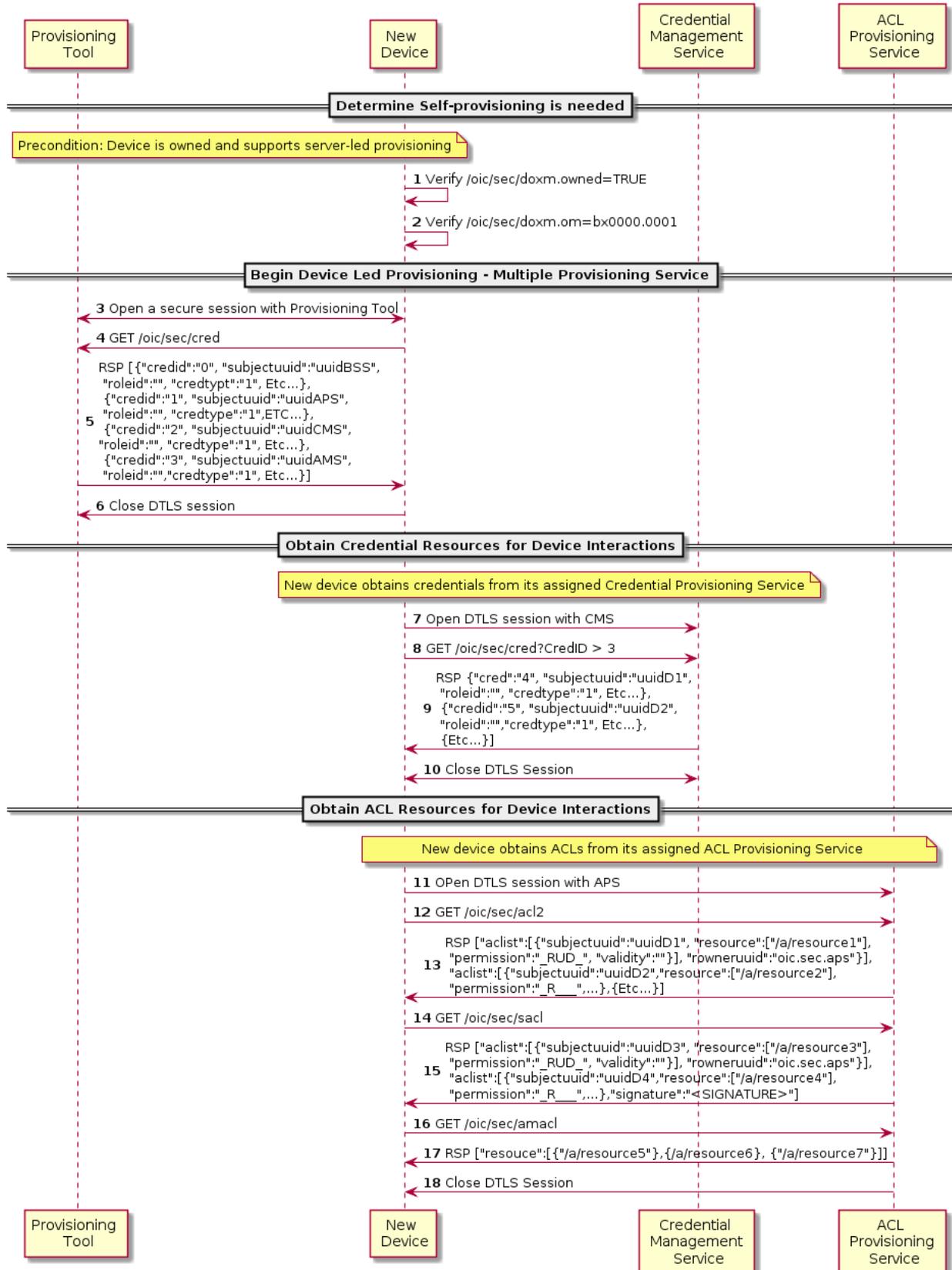
Step	Description
1	The new Device verifies it is owned.
2	The new Device verifies it is in self-provisioning mode.
3	The new Device verifies its target provisioning state is fully provisioned.
4	The new Device verifies its current provisioning state requires provisioning.
5	The new Device initiates a 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.

1981 **7.4.1.4 Server-directed Provisioning Involving Multiple Support Services**

1982 A Server-directed provisioning flow, involving multiple support services distributes the
 1983 provisioning work across multiple support services. Employing multiple support services is an
 1984 effective way to distribute provisioning workload or to deploy specialized support. The example in
 1985 Figure 26 demonstrates using a provisioning tool to configure two support services, a CMS and
 1986 an AMS. Steps for the example are described in Table 14.

OCF Server Led Provisioning with Multiple Service Providers



1988 **Figure 26 – Example of Server-directed provisioning involving multiple support services**

1989 **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.

1990 **7.5 Device Provisioning for OCF Cloud**

1991 **7.5.1 Cloud Provisioning General**

1992 The Device that connects to the OCF Cloud shall support the oic.r.coapcloudconf Resource on
 1993 Device and following SVRs on the OCF Cloud: "/oic/sec/account", "/oic/sec/session",
 1994 "/oic/sec/tokenrefresh".

1995 The OCF Cloud is expected to use a secure mechanism for associating a Mediator with an OCF
 1996 Cloud User. The choice of mechanism is up to the OCF Cloud. Example, mechanisms include
 1997 HTTP authentication (with username and password) or OAuth 2.0 (using an Authorization Server
 1998 which could be operated by the OCF Cloud provider or a third party). OCF Cloud is expected to
 1999 ensure that the suitable authentication mechanism is used to authenticate the OCF Cloud User.

2000 **7.5.2 Device Provisioning by Mediator**

2001 The Mediator and the Device shall use the secure session to provision the Device to connect with
 2002 the OCF Cloud.

2003 The Mediator obtains an Access Token from the OCF Cloud as described in ISO/IEC 30118-
 2004 8:2018. This Access Token is then used by the Device for registering with the OCF Cloud as

2005 described in 10.5. The OCF Cloud maintains a map where Access Token and Mediator provided
 2006 Device ID are stored. At the time of Device Registration OCF Cloud validates the Access Token
 2007 and associates the TLS session with corresponding Device ID.

2008 The Mediator provisions the Device, as described in ISO/IEC 30118-8:2018. The Mediator
 2009 provisions OCF Cloud URI to the "cis" Property of "oic.r.coapcloudconf" Resource, OCF Cloud
 2010 UUID to the "sid" Property of "oic.r.coapcloudconf" Resource and per-device Access Token to the
 2011 "at" Property of "oic.r.coapcloudconf" Resource on Device. Provisioned "at" is to be treated by
 2012 Device as an Access Token with "Bearer" token type as defined in IETF RFC 6750.

2013 For the purposes of access control, the Device shall identify the OCF Cloud using the OCF Cloud
 2014 UUID in the Common Name field of the End-Entity certificate used to authenticate the OCF Cloud.

2015 AMS should configure the ACE2 entries on a Device so that the Mediator(s) is the only Device(s)
 2016 with UPDATE permission for the oic.r.coapcloudconf Resource.

2017 The AMS should configure the ACE2 entries on the Device to allow request from the OCF Cloud.
 2018 By request from the Mediator, the AMS removes old ACL2 entries with previous OCF Cloud UUID.
 2019 This request happens before "oic.r.coapcloudconf" is configured by the Mediator for the new OCF
 2020 Cloud. The Mediator also requests AMS to set the OCF Cloud UUID as the "subject" Property for
 2021 the new ACL2 entries. AMS may use "sid" Property of "oic.r.coapcloudconf" Resource as the
 2022 current OCF Cloud UUID. AMS could either provision a wildcard entry for the OCF Cloud or
 2023 provision an entry listing each Resource published on the Device.

2024 If OCF Cloud provides "redirecturi" Value as response during Device Registration, the redirected-
 2025 to OCF Cloud is assumed to have the same OCF Cloud UUID and to use the same trust anchor.
 2026 Otherwise, presented OCF Cloud UUID wouldn't match the provisioned ACL2 entries.

2027 The Mediator should provision the oic.r.coapcloudconf Resource with the Properties in Table 15.
 2028 These details once provisioned are used by the Device to perform Device Registration to the
 2029 OCF Cloud. After the initial registration, the Device should use updated values received from the
 2030 OCF Cloud instead. If OCF Cloud User wants the Device to re-register with the OCF Cloud, they
 2031 can use the Mediator to re-provision the oic.r.coapcloudconf Resource with the new values.

2032 **Table 15 – Mapping of Properties of the oic.r.account and oic.r.coapcloudconf Resources**

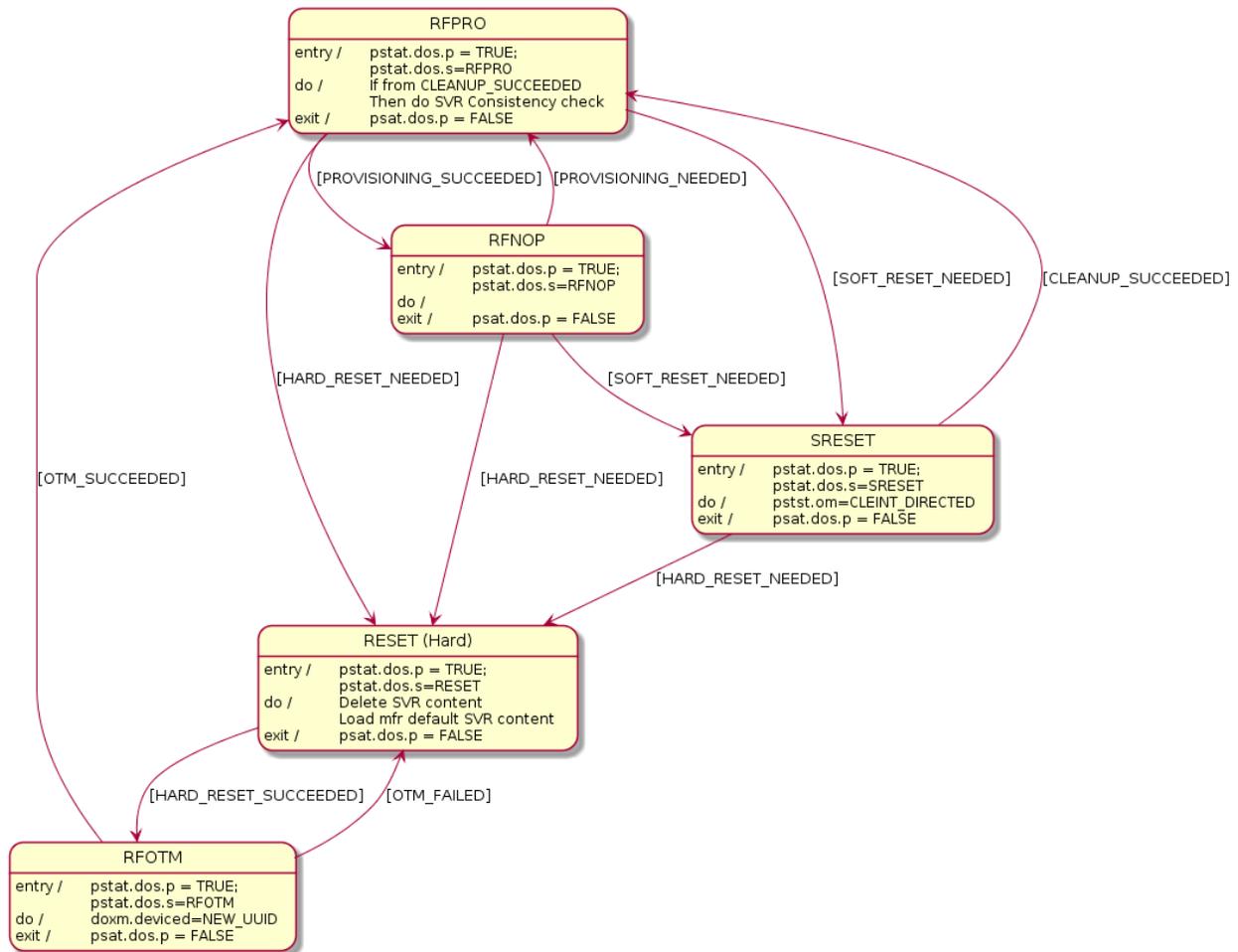
Property Name	oic.r.coapcloudconf	oic.r.account	Description
Authorization Provider Name	apn	authprovider	The Authorization Provider through which Access Token was obtained.
OCF Cloud URL	cis	-	This is the URL connection is established between Device and OCF Cloud.
Access Token	at	accesstoken	The unique 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.

2033 **8 Device Onboarding State Definitions**

2034 **8.1 Device Onboarding General**

2035 As explained in 5.3, the process of onboarding completes after the ownership of the Device has
 2036 been transferred and the Device has been provisioned with relevant configuration/services as
 2037 explained in 5.4.. The Figure 27 shows the various states a Device can be in during the Device
 2038 lifecycle.

2039 The /pstat.dos.s Property is RW by the /oic/sec/pstat resource owner (e.g. "doxs" service) so that
 2040 the resource owner can remotely update the Device state. When the Device is in RFNOP or
 2041 RFPRO, ACLs can be used to allow remote control of Device state by other Devices. When the
 2042 Device state is SRESET the Device OC may be the only indication of authorization to access the
 2043 Device. The Device owner may perform low-level consistency checks and re-provisioning to get
 2044 the Device suitable for a transition to RFPRO.



2045
 2046 **Figure 27 – Device state model**

2047 As shown in the diagram, at the conclusion of the provisioning step, the Device comes in the
 2048 "Ready for Normal Operation" state where it has all it needs in order to start interoperating with
 2049 other Devices. 8.2 specifies the minimum mandatory configuration that a Device shall hold in
 2050 order to be considered as "Ready for Normal Operation".

2051 In the event of power loss or Device failure, the Device should remain in the same state that it
 2052 was in prior to the power loss / failure

2053 If a Device or resource owner OBSERVES /pstat.dos.s, then transitions to SRESET will give early
 2054 warning notification of Devices that may require SVR consistency checking.

2055 In order for onboarding to function, the Device shall have the following Resources installed:

- 2056 1) "/oic/sec/doxm" Resource
 2057 2) "/oic/sec/pstat" Resource

2058 3) "/oic/sec/cred" Resource

2059 The values contained in these Resources are specified in the state definitions in 8.2, 8.3, 8.4, 8.5
2060 and 8.6.

2061 **8.2 Device Onboarding-Reset State Definition**

2062 The /pstat.dos.s = RESET state is defined as a "hard" reset to manufacturer defaults. Hard reset
2063 also defines a state where the Device asset is ready to be transferred to another party.

2064 The Platform manufacturer should provide a physical mechanism (e.g. button) that forces
2065 Platform reset. All Devices hosted on the same Platform transition their Device states to RESET
2066 when the Platform reset is asserted.

2067 The following Resources and their specific properties shall have the value as specified:

2068 1) The owned Property of the "/oic/sec/doxm" Resource shall transition to FALSE.

2069 2) The devowneruuid Property of the "/oic/sec/doxm" Resource shall be nil UUID.

2070 3) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
2071 implemented.

2072 4) The deviceuuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
2073 default value.

2074 5) The deviceid Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
2075 default value, if this Property is implemented.

2076 6) The sct Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's default
2077 value.

2078 7) The oxmsel Property of the "/oic/sec/doxm" Resource shall be reset to the manufacturer's
2079 default value.

2080 8) The isop Property of the "/oic/sec/pstat" Resource shall be FALSE.

2081 9) The dos Property of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal
2082 "RESET" state and dos.p shall equal "FALSE".

2083 10)

2084 11) The om (operational modes) Property of the "/oic/sec/pstat" Resource shall be set to the
2085 manufacturer default value.

2086 12) The sm (supported operational modes) Property of the /oic/sec/pstat Resource shall be set to
2087 the manufacturer default value.

2088 13) The rowneruuid Property of "/oic/sec/pstat", "/oic/sec/doxm", "/oic/sec/acl", "/oic/sec/amacl",
2089 "/oic/sec/sacl", and "/oic/sec/cred" Resources shall be nil UUID.

2090 14) The supportedprofiles Property of the /oic/sec/sp Resource shall be set to the manufacturer
2091 default value.

2092 15) The currentprofile Property of the /oic/sec/sp Resource shall be set to the manufacturer
2093 default value.

2094 **8.3 Device Ready-for-OTM State Definition**

2095 The following Resources and their specific properties shall have the value as specified when the
2096 Device enters ready for ownership transfer:

2097 1) The owned Property of the "/oic/sec/doxm" Resource shall be FALSE and will transition to
2098 TRUE.

2099 2) The devowner Property of the "/oic/sec/doxm" Resource shall be nil UUID, if this Property is
2100 implemented.

- 2101 3) The devowneruuid Property of the "/oic/sec/doxm" Resource shall be nil UUID.
- 2102 4) The deviceid Property of the "/oic/sec/doxm" Resource may be nil UUID, if this Property is
2103 implemented. The value of the di Property in /oic/d is undefined.
- 2104 5) The deviceuuid Property of the "/oic/sec/doxm" Resource shall be set to the manufacturer
2105 default value.
- 2106 6) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
- 2107 7) The dos of the "/oic/sec/pstat" Resource shall be updated: dos.s shall equal "RFOTM" state
2108 and dos.p shall equal "FALSE".
- 2109 8) The "/oic/sec/cred" Resource shall contain credential(s) if required by the selected OTM

2110 **8.4 Device Ready-for-Provisioning State Definition**

2111 The following Resources and their specific properties shall have the value as specified when the
2112 Device enters ready for provisioning:

- 2113 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
- 2114 2) The devowneruuid Property of the /oic/sec/doxm Resource shall not be nil UUID.
- 2115 3) The deviceuuid Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be set
2116 to the value that was determined during RFOTM processing. Also the value of the di Property
2117 in /oic/d Resource shall be the same as the deviceid Property in the /oic/sec/doxm Resource.
- 2118 4) The oxmsel Property of the /oic/sec/doxm Resource shall have the value of the actual OTM
2119 used during ownership transfer.
- 2120 5) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
- 2121 6) The dos of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFPRO" state
2122 and dos.p shall equal "FALSE".
- 2123 7) The rowneruuid Property of every installed Resource shall be set to a valid Resource owner
2124 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
2125 rowneruuid may result in an orphan Resource.
- 2126 8) The /oic/sec/cred Resource shall contain credentials for each entity referenced by an
2127 rowneruuid, amsuuid, devowneruuid.

2128 **8.5 Device Ready-for-Normal-Operation State Definition**

2129 The following Resources and their specific properties shall have the value as specified when the
2130 Device enters ready for normal operation:

- 2131 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
- 2132 2) The devowneruuid Property of the /oic/sec/doxm Resource shall not be nil UUID.
- 2133 3) The deviceuuid Property of the /oic/sec/doxm Resource shall not be nil UUID and shall be set
2134 to the ID that was configured during OTM. Also the value of the "di" Property in /oic/d shall be
2135 the same as the deviceuuid.
- 2136 4) The oxmsel Property of the /oic/sec/doxm Resource shall have the value of the actual OTM
2137 used during ownership transfer.
- 2138 5) The isop Property of the /oic/sec/pstat Resource shall be set to TRUE by the Server once
2139 transition to RFNOP is otherwise complete.
- 2140 6) The dos of the /oic/sec/pstat Resource shall be updated: dos.s shall equal "RFNOP" state
2141 and dos.p shall equal "FALSE".
- 2142 7) The rowneruuid Property of every installed Resource shall be set to a valid resource owner
2143 (i.e. an entity that is authorized to instantiate or update the given Resource). Failure to set a
2144 rowneruuid results in an orphan Resource.

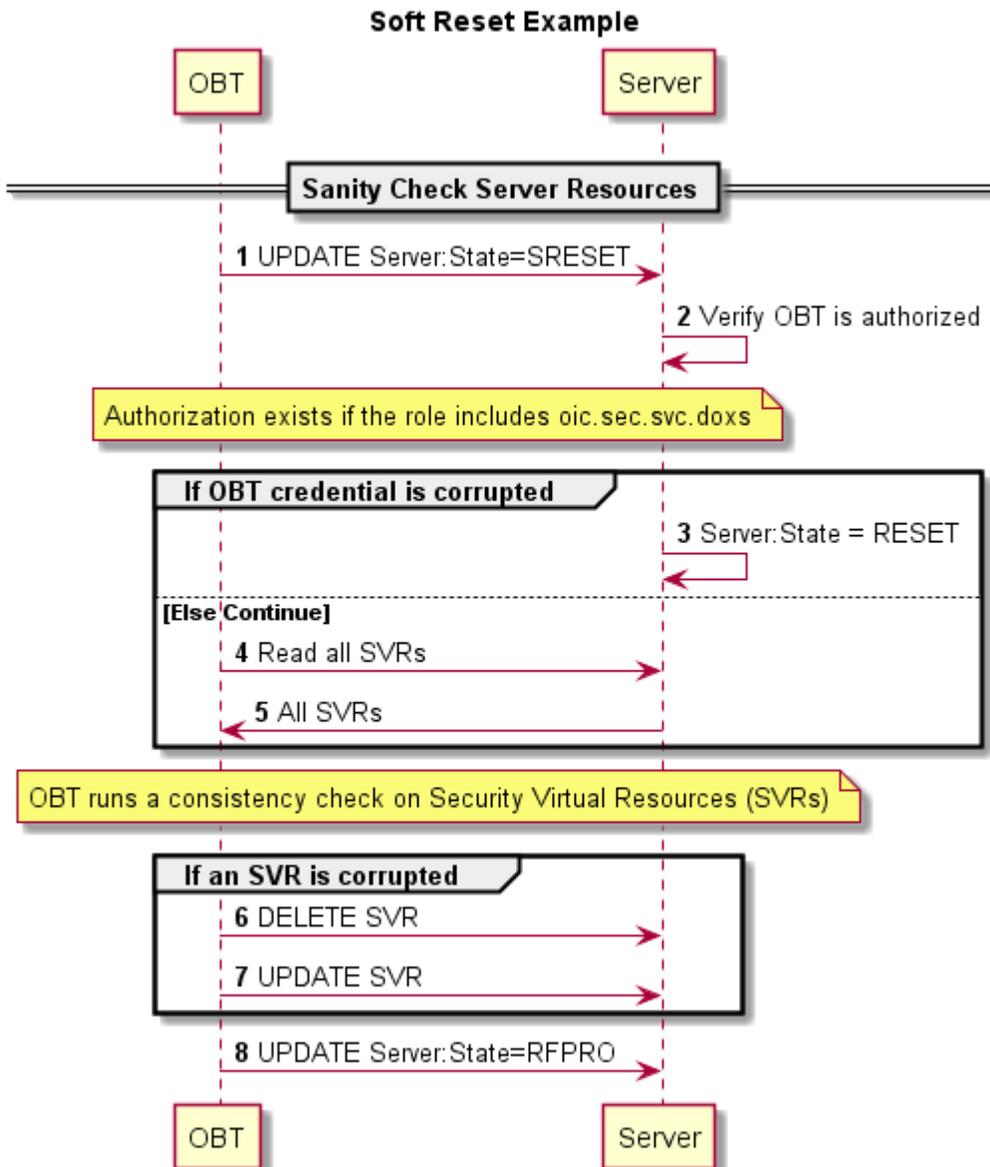
2145 8) The /oic/sec/cred Resource shall contain credentials for each service referenced by a
 2146 rowneruuid, amsuuid, devowneruuid.

2147 **8.6 Device Soft Reset State Definition**

2148 The soft reset state is defined (e.g. /pstat.dos.s = SRESET) where entrance into this state means
 2149 the Device is not operational but remains owned by the current owner. The Device may exit
 2150 SRESET by authenticating to a DOTS (e.g. "rt" = "oic.r.doxs") using the OC provided during
 2151 original onboarding (but should not require use of an OTM /doxm.oxms).

2152 The DOTS should perform a consistency check of the SVR and if necessary, re-provision them
 2153 sufficiently to allow the Device to transition to RFPRO.

2154 Figure 28 depicts OBT Sanity Check Sequence in SRESET.



2155

2156

Figure 28 – OBT Sanity Check Sequence in SRESET

2157 The DOTS should perform a sanity check of SVRs before final transition to RFPRO Device state.
2158 If the DOTS credential cannot be found or is determined to be corrupted, the Device state
2159 transitions to RESET. The Device should remain in SRESET if the DOTS credential fails to
2160 validate the DOTS. This mitigates denial-of-service attacks that may be attempted by non-DOTS
2161 Devices.

2162 When in SRESET, the following Resources and their specific Properties shall have the values as
2163 specified.

- 2164 1) The owned Property of the /oic/sec/doxm Resource shall be TRUE.
 - 2165 2) The devowneruuid Property of the /oic/sec/doxm Resource shall remain non-null.
 - 2166 3) The devowner Property of the /oic/sec/doxm Resource shall be non-null, if this Property is
2167 implemented.
 - 2168 4) The deviceuuidProperty of the /oic/sec/doxm Resource shall remain non-null.
 - 2169 5) The deviceid Property of the /oic/sec/doxm Resource shall remain non-null.
 - 2170 6) The sct Property of the /oic/sec/doxm Resource shall retain its value.
 - 2171 7) The oxmsel Property of the /oic/sec/doxm Resource shall retains its value.
 - 2172 8) The isop Property of the /oic/sec/pstat Resource shall be FALSE.
 - 2173 9) The /oic/sec/pstat.dos.s Property shall be SRESET.
 - 2174 10) The om (operational modes) Property of the /oic/sec/pstat Resource shall be "client-directed
2175 mode".
 - 2176 11) The sm (supported operational modes) Property of /oic/sec/pstat Resource may be updated
2177 by the Device owner (aka DOTS).
 - 2178 12) The rowneruuid Property of /oic/sec/pstat, /oic/sec/doxm, /oic/sec/acl, /oic/sec/acl2,
2179 /oic/sec/amacl, /oic/sec/sacl, and /oic/sec/cred Resources may be reset by the Device owner
2180 (aka DOTS) and re-provisioned.
- 2181

2182 **9 Security Credential Management**

2183 **9.1 Preamble**

2184 This clause provides an overview of the credential types in OCF, along with details of credential
2185 use, provisioning and ongoing management.

2186 **9.2 Credential Lifecycle**

2187 **9.2.1 Credential Lifecycle General**

2188 OCF credential lifecycle has the following phases: (1) creation, (2) deletion, (3) refresh, (4)
2189 issuance and (5) revocation.

2190 **9.2.2 Creation**

2191 The CMS shall provision credential Resources to the Device. The Device shall verify the CMS is
2192 authorized by matching the rowneruuid Property of the /oic/sec/cred resource to the DeviceID of
2193 the credential the CMS used to establish the secure connection.

2194 Credential Resources created using a CMS may involve specialized credential issuance protocols
2195 and messages. These may involve the use of public key infrastructure (PKI) such as a certificate
2196 authority (CA), symmetric key management such as a key distribution centre (KDC) or as part of
2197 a provisioning action by a DOTS, CMS or AMS.

2198 **9.2.3 Deletion**

2199 The CMS should delete known compromised credential Resources. The Device (e.g. the Device
2200 where the credential Resource is hosted) should delete credential Resources that have expired.

2201 An expired credential Resource may be deleted to manage memory and storage space.

2202 Deletion in OCF key management is equivalent to credential suspension.

2203 **9.2.4 Refresh**

2204 Credential refresh may be performed before it expires. The CMS shall perform credential refresh.

2205 The /oic/sec/cred Resource supports expiry using the Period Property. Credential refresh may be
2206 applied when a credential is about to expire or is about to exceed a maximum threshold for bytes
2207 encrypted.

2208 A credential refresh method specifies the options available when performing key refresh. The
2209 Period Property informs when the credential should expire. The Device may proactively obtain a
2210 new credential using a credential refresh method using current unexpired credentials to refresh
2211 the existing credential. If the Device does not have an internal time source, the current time
2212 should be obtained from a CMS at regular intervals.

2213 If the CMS credential is allowed to expire, the DOTS service may be used to re-provision the
2214 CMS credentials to the Device. If the onboarding established credentials are allowed to expire
2215 the DOTS shall re-onboard the Device to re-apply device owner transfer steps.

2216 All Devices shall support at least one credential refresh method.

2217 **9.2.5 Revocation**

2218 Credentials issued by a CMS may be equipped with revocation capabilities. In situations where
2219 the revocation method involves provisioning of a revocation object that identifies a credential that
2220 is to be revoked prior to its normal expiration period, a credential Resource is created containing
2221 the revocation information that supersedes the originally issued credential. The revocation object

2222 expiration should match that of the revoked credential so that the revocation object is cleaned up
2223 upon expiry.

2224 It is conceptually reasonable to consider revocation applying to a credential or to a Device.
2225 Device revocation asserts all credentials associated with the revoked Device should be
2226 considered for revocation. Device revocation is necessary when a Device is lost, stolen or
2227 compromised. Deletion of credentials on a revoked Device might not be possible or reliable.

2228 **9.3 Credential Types**

2229 **9.3.1 Preamble**

2230 The “/oic/sec/cred” Resource maintains a credential type Property that supports several
2231 cryptographic keys and other information used for authentication and data protection. The
2232 credential types supported include pair-wise symmetric keys, group symmetric keys, asymmetric
2233 authentication keys, certificates (i.e. signed asymmetric keys) and shared-secrets (i.e.
2234 PIN/password).

2235 **9.3.2 Pair-wise Symmetric Key Credentials**

2236 The CMS shall provision exactly one other pair-wise symmetric credential to a peer Device. The
2237 CMS should not store pair-wise symmetric keys it provisions to managed Devices.

2238 Pair-wise keys could be established through ad-hoc key agreement protocols.

2239 The PrivateData Property in the “/oic/sec/cred” Resource contains the symmetric key.

2240 The PublicData Property may contain a token encrypted to the peer Device containing the pair-
2241 wise key.

2242 The OptionalData Property may contain revocation status.

2243 The Device implementer should apply hardened key storage techniques that ensure the
2244 PrivateData remains private.

2245 The Device implementer should apply appropriate integrity, confidentiality and access protection
2246 of the “/oic/sec/cred”, “/oic/sec/crl”, “/oic/sec/roles”, “/oic/sec/csr” Resources to prevent
2247 unauthorized modifications.

2248 **9.3.3 Group Symmetric Key Credentials**

2249 Group keys are symmetric keys shared among a group of Devices (3 or more). Group keys are
2250 used for efficient sharing of data among group participants.

2251 Group keys do not provide authentication of Devices but only establish membership in a group.

2252 The CMS shall provision group symmetric key credentials to the group members. The CMS
2253 maintains the group memberships.

2254 The PrivateData Property in the “/oic/sec/cred” Resource contains the symmetric key.

2255 The PublicData Property may contain the group name.

2256 The OptionalData Property may contain revocation status.

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

2259 The Device implementer should apply appropriate integrity, confidentiality and access protection
2260 of the “/oic/sec/cred”, “/oic/sec/crl”, “/oic/sec/roles”, “/oic/sec/csr” Resources to prevent
2261 unauthorized modifications.

2262 **9.3.4 Asymmetric Authentication Key Credentials**

2263 **9.3.4.1 Asymmetric Authentication Key Credentials General**

2264 Asymmetric authentication key credentials contain either a public and private key pair or only a
2265 public key. The private key is used to sign Device authentication challenges. The public key is
2266 used to verify a device authentication challenge-response.

2267 The PrivateData Property in the “/oic/sec/cred” Resource contains the private key.

2268 The PublicData Property contains the public key.

2269 The OptionalData Property may contain revocation status.

2270 The Device implementer should apply hardened key storage techniques that ensure the
2271 PrivateData remains private.

2272 Devices should generate asymmetric authentication key pairs internally to ensure the private key
2273 is only known by the Device. See 9.3.4.2 for when it is necessary to transport private key material
2274 between Devices.

2275 The Device implementer should apply appropriate integrity, confidentiality and access protection
2276 of the “/oic/sec/cred”, “/oic/sec/crl”, “/oic/sec/roles”, “/oic/sec/csr” Resources to prevent
2277 unauthorized modifications.

2278 **9.3.4.2 External Creation of Asymmetric Authentication Key Credentials**

2279 Devices should employ industry-standard high-assurance techniques when allowing off-device
2280 key pair creation and provisioning. Use of such key pairs should be minimized, particularly if the
2281 key pair is immutable and cannot be changed or replaced after provisioning.

2282 When used as part of onboarding, these key pairs can be used to prove the Device possesses
2283 the manufacturer-asserted properties in a certificate to convince a DOTS or a user to accept
2284 onboarding the Device. See 7.3.3 for the OTM that uses such a certificate to authenticate the
2285 Device, and then provisions new OCF Security Domain credentials for use.

2286 **9.3.5 Asymmetric Key Encryption Key Credentials**

2287 The asymmetric key-encryption-key (KEK) credentials are used to wrap symmetric keys when
2288 distributing or storing the key.

2289 The PrivateData Property in the “/oic/sec/cred” Resource contains the private key.

2290 The PublicData Property contains the public key.

2291 The OptionalData Property may contain revocation status.

2292 The Device implementer should apply hardened key storage techniques that ensure the
2293 PrivateData remains private.

2294 The Device implementer should apply appropriate integrity, confidentiality and access protection
2295 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
2296 modifications.

2297 **9.3.6 Certificate Credentials**

2298 Certificate credentials are asymmetric keys that are accompanied by a certificate issued by a
2299 CMS or an external certificate authority (CA).

2300 A certificate enrolment protocol is used to obtain a certificate and establish proof-of-possession.

2301 The issued certificate is stored with the asymmetric key credential Resource.

2302 Other objects useful in managing certificate lifecycle such as certificate revocation status are
2303 associated with the credential Resource.

2304 Either an asymmetric key credential Resource or a self-signed certificate credential is used to
2305 terminate a path validation.

2306 The PrivateData Property in the /oic/sec/cred Resource contains the private key.

2307 The PublicData Property contains the issued certificate.

2308 The OptionalData Property may contain revocation status.

2309 The Device implementer should apply hardened key storage techniques that ensure the
2310 PrivateData remains private.

2311 The Device implementer should apply appropriate integrity, confidentiality and access protection
2312 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
2313 modifications.

2314 **9.3.7 Password Credentials**

2315 Shared secret credentials are used to maintain a PIN or password that authorizes Device access
2316 to a foreign system or Device that doesn't support any other OCF credential types.

2317 The PrivateData Property in the /oic/sec/cred Resource contains the PIN, password and other
2318 values useful for changing and verifying the password.

2319 The PublicData Property may contain the user or account name if applicable.

2320 The OptionalData Property may contain revocation status.

2321 The Device implementer should apply hardened key storage techniques that ensure the
2322 PrivateData remains private.

2323 The Device implementer should apply appropriate integrity, confidentiality and access protection
2324 of the /oic/sec/cred, /oic/sec/crl, /oic/sec/roles, /oic/sec/csr Resources to prevent unauthorized
2325 modifications.

2326 **9.4 Certificate Based Key Management**

2327 **9.4.1 Overview**

2328 To achieve authentication and transport security during communications in OCF Security Domain,
2329 certificates containing public keys of communicating parties and private keys can be used.

2330 The certificate and private key may be issued by a local or remote certificate authority (CA). For
2331 the local CA, a certificate revocation list (CRL) based on X.509 is used to validate proof of
2332 identity. In the case of a remote CA, Online Certificate Status Protocol (OCSP) can be used to
2333 validate proof of identity and validity.

2334 The OCF certificate and OCF CRL (Certificate Revocation List) format is a subset of X.509 format,
2335 only elliptic curve algorithm and DER encoding format are allowed, most of optional fields in
2336 X.509 are not supported so that the format intends to meet the constrained Device's requirement.

2337 As for the certificate and CRL management in the Server, the process of storing, retrieving and
2338 parsing Resources of the certificates and CRL will be performed at the security resource
2339 manager layer; the relevant interfaces may be exposed to the upper layer.

2340 A SRM is the security enforcement point in a Server as described in clause 5.5, so the data of
2341 certificates and CRL will be stored and managed in SVR database.

2342 The CMS manages the certificate lifecycle for certificates it issues. The DOTS shall assign a
2343 CMS to a Device when it is newly onboarded. The issuing CMS should process certificate
2344 revocations for certificates it issues. If a certificate private key is compromised, the CMS should
2345 revoke the certificate. If CRLs are used by a Device, the CMS should regularly (for example;
2346 every 3 months) update the /oic/sec/crl resource for the Devices it manages.

2347 **9.4.2 X.509 Digital Certificate Profiles**

2348 **9.4.2.1 Digital Certificate Profile General**

2349 An OCF certificate format is a subset of X.509 format (version 3 or above) as defined in
2350 IETF RFC 5280.

2351 This clause develops a profile to facilitate the use of X.509 certificates within OCF applications
2352 for those communities wishing to make use of X.509 technology. The X.509 v3 certificate format
2353 is described in detail, with additional information regarding the format and semantics of OCF
2354 specific extension(s). The supported standard certificate extensions are also listed.

2355 Certificate Format: The OCF certificate profile is derived from IETF RFC 5280. However, this
2356 document does not support the "issuerUniqueID" and "subjectUniqueID" fields which are
2357 deprecated and shall not be used in the context of OCF. If these fields are present in a certificate,
2358 compliant entities shall ignore their contents.

2359 Certificate Encoding: Conforming entities shall use the Distinguished Encoding Rules (DER) as
2360 defined in ISO/IEC 8825-1 to encode certificates.

2361 Certificates Hierarchy and Crypto Parameters. OCF supports a three-tier hierarchy for its Public
2362 Key Infrastructure (i.e., a Root CA, an Intermediate CA, and EE certificates). OCF accredited CAs
2363 SHALL use Elliptic Curve Cryptography (ECC) keys (secp256r1 – OID:1.2.840.10045.3.1.7) and
2364 use the ecdsaWithSHA256 (OID:1.2.840.10045.4.3.2) algorithm for certificate signatures.

2365 The following clauses specify the supported standard and custom extensions for the OCF
2366 certificates profile.

2367 **9.4.2.2 Certificate Profile and Fields**

2368 **9.4.2.2.1 Root CA Certificate Profile**

2369 Table 16 describes X.509 v1 fields required for Root CA Certificates.

2370 **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)

2371 **Table 17** describes X.509 v3 extensions required for Root CA Certificates.

2372 **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)

2373 **9.4.2.2.2 Intermediate CA Certificate Profile**

2374 Table 18 describes X.509 v1 fields required for Intermediate CA Certificates.

2375 **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)

2376 **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

2378 **9.4.2.2.3 End-Entity Black Certificate Profile**

2379 Table 20 describes X.509 v1 fields required for End-Entity Certificates used for Black security
2380 profile.

2381

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)

2382 **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
authorityKeyIdIdentifier	OPTIONAL	Non-critical	N/A
subjectKeyIdIdentifier	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"> • serverAuthentication - 1.3.6.1.5.5.7.3.1 • clientAuthentication - 1.3.6.1.5.5.7.3.2 Exactly ONE of the following OIDs SHALL be present: <ul style="list-style-type: none"> • Identity certificate - 1.3.6.1.4.1.44924.1.6 • Role certificate - 1.3.6.1.4.1.44924.1.7 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

2384 **9.4.2.2.4 OCF Compliance X.509v3 Extension**

2385 The OCF Compliance Extension defines required parameters to correctly identify the type of
2386 Device, its manufacturer, its OCF Version, and the Security Profile compliance of the device.

2387 The extension carries an "ocfVersion" field which provides the specific base version of the OCF
2388 documents the device implements. The "ocfVersion" field shall contain a sequence of three
2389 integers ("major", "minor", and "build"). For example, if an entity is certified to be compliant with

2390 OCF specifications 1.3.2, then the "major", "minor", and "build" fields of the "ocfVersion" will be
2391 set to "1", "3", and "2" respectively. The "ocfVersion" may be used by Security Profiles to denote
2392 compliance to a specified base version of the OCF documents.

2393 The 'securityProfile' field shall carry the ocfSecurityProfile OID(s) (clause 14.8.3) of one or more
2394 supported Security Profiles associated with the certificate in string form (UTF-8). All Security
2395 Profiles associated with the certificate should be identified by this field.

2396 The extension shall also carry two string fields (UTF-8): "DeviceName" and "deviceManufacturer".
2397 The fields carry human-readable descriptions of the Device's name and manufacturer,
2398 respectively.

2399 The ASN.1 definition of the OCFCCompliance extension (OID – 1.3.6.1.4.1.51414.1.0) is defined
2400 as follows:

```
2401 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2402                               private(4) enterprise(1) OCF(51414) }
2403
2404 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2405
2406 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
2407
2408 ocfVersion ::= SEQUENCE {
2409     major    INTEGER,
2410             --Major version number
2411     minor    INTEGER,
2412             --Minor version number
2413     build    INTEGER,
2414             --Build/Micro version number
2415 }
2416
2417 ocfCompliance ::= SEQUENCE {
2418     version          ocfVersion,
2419                    --Device/OCF version
2420     securityProfile  SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
2421                    --Sequence of OCF Security Profile OID strings
2422                    --Clause 14.8.2 defines valid ocfSecurityProfileOIDs
2423     deviceName       UTF8String,
2424                    --Name of the device
2425     deviceManufacturer UTF8String,
2426                    --Human-Readable Manufacturer
2427                    --of the device
2428 }
```

2429 **9.4.2.2.5 Manufacturer Usage Description (MUD) X.509v3 Extension**

2430 The goal of the Manufacturer Usage Description (MUD) extension is to provide a means for
2431 devices to signal to the network the access and network functionality they require to properly
2432 function. Access controls can be more easily achieved and deployed at scale when the MUD
2433 extension is used. The current draft of the MUD v3 extension at this time of writing is:

2434 <https://tools.ietf.org/html/draft-ietf-opsawg-mud-15#section-10>

2435 The ASN.1 definition of the MUD v3 extension is defined as follows:

```
2436 MUDURLExtnModule-2016 { iso(1) identified-organization(3) dod(6)
2437                          internet(1) security(5) mechanisms(5) pkix(7)
2438                          id-mod(0) id-mod-mudURLExtn2016(88) }
2439
2440 DEFINITIONS IMPLICIT TAGS ::= BEGIN
2441 -- EXPORTS ALL --
2442 IMPORTS
```

```

2443     EXTENSION
2444     FROM PKIX-CommonTypes-2009
2445         { iso(1) identified-organization(3) dod(6) internet(1)
2446           security(5) mechanisms(5) pkix(7) id-mod(0)
2447           id-mod-pkixCommon-02(57) }
2448     id-pe
2449     FROM PKIX1Explicit-2009
2450         { iso(1) identified-organization(3) dod(6) internet(1)
2451           security(5) mechanisms(5) pkix(7) id-mod(0)
2452           id-mod-pkix1-explicit-02(51) } ;
2453     MUDCertExtensions EXTENSION ::= { ext-MUDURL, ... }
2454     ext-MUDURL EXTENSION ::= { SYNTAX MUDURLSyntax
2455                               IDENTIFIED BY id-pe-mud-url }
2456
2457     id-pe-mud-url OBJECT IDENTIFIER ::= { id-pe 25 }
2458
2459     MUDURLSyntax ::= IA5String
2460
2461     END

```

2462 **9.4.2.2.6 OCF Security Claims X.509v3 Extension**

2463 The OCF Security Claims Extension defines a list of OIDs representing security claims that the
2464 manufacturer/integrator is making as to the security posture of the device above those required
2465 by the OCF Compliance version or that of the OCF Security Profile being indicated by the device.

2466 The purpose of this extension is to allow for programmatic evaluation of assertions made about
2467 security to enable some platforms/policies/administrators to better understand what is being
2468 onboarded or challenged.

2469 The ASN.1 definition of the OCF Security Claims extension (OID – 1.3.6.1.4.1.51414.1.1) is
2470 defined as follows:

```

2471 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2472                               private(4) enterprise(1) OCF(51414) }
2473
2474 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2475
2476 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
2477
2478     claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
2479     --Device claims that the boot process follows a procedure trusted
2480     --by the firmware and the BIOS
2481
2482     claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
2483     --Device claims that credentials are stored in a specialized hardware
2484     --protection environment such as a Trusted Platform Module (TPM) or
2485     --similar mechanism.
2486
2487     ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
2488
2489     ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID

```

2490 **9.4.2.2.7 OCF Certified Product List Attributes X.509v3 Extension**

2491 The OCF Certified Product List Extension defines required parameters to utilize the OCF
2492 Compliance Management System Certified Product List (OCMS-CPL). This clause is only
2493 applicable if you plan to utilize the OCMS-CPL. The OBT may make use of these attributes to
2494 verify the compliance level of a device.

2495 The extension carries the OCF CPL Attributes: IANA Private Enterprise Number (PEN), Model
2496 and Version.

2497 The 'cpl-at-IANAPen' IANA Private Enterprise Number (PEN) provides the manufacturer's unique
2498 PEN established in the IANA PEN list located at: [https://www.iana.org/enterprise-](https://www.iana.org/assignments/enterprise-)
2499 numbers. The 'cpl-at-IANAPen' field found in end-products shall be the same information as
2500 reported during OCF Certification.

2501 The 'cpl-at-model' represents an OCF-Certified product's model name. The 'cpl-at-model' field
2502 found in end-products shall be the same information as reported during OCF Certification.

2503 The 'cpl-at-version' represents an OCF-Certified product's version. The 'cpl-at-version' field found
2504 in end-products shall be the same information as reported during OCF Certification.

2505 The ASN.1 definition of the OCF CPL Attributes extension (OID – 1.3.6.1.4.1.51414.1.2) is
2506 defined as follows:

```
2507 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
2508                               private(4) enterprise(1) OCF(51414) }
2509
2510 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
2511
2512     id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
2513
2514         cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
2515         cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
2516         cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
2517
2518
2519     ocfCPLAttributes ::= SEQUENCE {
2520         cpl-at-IANAPen      UTF8String,
2521         --Manufacturer's registered IANA Private Enterprise Number
2522         cpl-at-model      UTF8String,
2523         --Device OCF Security Profile
2524         cpl-at-version    UTF8String
2525         --Name of the device
2526     }
```

2527 9.4.2.3 Supported Certificate Extensions

2528 As these certificate extensions are a standard part of IETF RFC 5280, this document includes the
2529 clause number from that RFC to include it by reference. Each extension is summarized here, and
2530 any modifications to the RFC definition are listed. Devices MUST implement and understand the
2531 extensions listed here; other extensions from the RFC are not included in this document and
2532 therefore are not required. 10.4 describes what Devices must implement when validating
2533 certificate chains, including processing of extensions, and actions to take when certain
2534 extensions are absent.

2535 – Authority Key Identifier (4.2.1.1)

2536 The Authority Key Identifier (AKI) extension provides a means of identifying the public key
2537 corresponding to the private key used to sign a certificate. This document makes the
2538 following modifications to the referenced definition of this extension:

2539 The authorityCertIssuer or authorityCertSerialNumber fields of the AuthorityKeyIdentifier
2540 sequence are not permitted; only keyIdentifier is allowed. This results in the following
2541 grammar definition:

```
2542 id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }
2543
2544 AuthorityKeyIdentifier ::= SEQUENCE {
2545     keyIdentifier          [0] KeyIdentifier
2546 }
2547 KeyIdentifier ::= OCTET STRING
```

2548 – Subject Key Identifier (4.2.1.2)

2549 The Subject Key Identifier (SKI) extension provides a means of identifying certificates that
2550 contain a particular public key.

2551 This document makes the following modification to the referenced definition of this extension:

2552 Subject Key Identifiers SHOULD be derived from the public key contained in the certificate's
2553 SubjectPublicKeyInfo field or a method that generates unique values. This document
2554 RECOMMENDS the 256-bit SHA-2 hash of the value of the BIT STRING subjectPublicKey
2555 (excluding the tag, length, and number of unused bits). Devices verifying certificate chains
2556 must not assume any particular method of computing key identifiers, however, and must only
2557 base matching AKI's and SKI's in certification path constructions on key identifiers seen in
2558 certificates.

2559 – Subject Alternative Name

2560 If the EKU extension is present, and has the value XXXXXX, indicating that this is a role
2561 certificate, the Subject Alternative Name (subjectAltName) extension shall be present and
2562 interpreted as described below. When no EKU is present, or has another value, the
2563 subjectAltName extension SHOULD be absent. The subjectAltName extension is used to
2564 encode one or more Role ID values in role certificates, binding the roles to the subject public
2565 key. The subjectAltName extension is defined in IETF RFC 5280 (See 4.2.1.6):

```
2566 id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }
2567
2568 SubjectAltName ::= GeneralNames
2569
2570 GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
2571
2572 GeneralName ::= CHOICE {
2573     otherName                [0]     OtherName,
2574     rfc5322Name              [1]     IA5String,
2575     dNSName                  [2]     IA5String,
2576     x400Address              [3]     ORAddress,
2577     directoryName            [4]     Name,
2578     ediPartyName             [5]     EDIPartyName,
2579     uniformResourceIdentifier [6]     IA5String,
2580     iPAddress                [7]     OCTET STRING,
2581     registeredID             [8]     OBJECT IDENTIFIER }
2582
2583     EDIPartyName ::= SEQUENCE {
2584         nameAssigner          [0]     DirectoryString OPTIONAL,
2585         partyName             [1]     DirectoryString }
2586
```

2587 Each GeneralName in the GeneralNames SEQUENCE which encodes a role shall be a
2588 directoryName, which is of type Name. Name is an X.501 Distinguished Name. Each Name
2589 shall contain exactly one CN (Common Name) component, and zero or one OU
2590 (Organizational Unit) components. The OU component, if present, shall specify the authority
2591 that defined the semantics of the role. If the OU component is absent, the certificate issuer
2592 has defined the role. The CN component shall encode the role ID. Other GeneralName types
2593 in the SEQUENCE may be present, but shall not be interpreted as roles. Therefore, if the
2594 certificate issuer includes non-role names in the subjectAltName extension, the extension
2595 should not be marked critical.

2596 The role, and authority need to be encoded as ASN.1 PrintableString type, the restricted
2597 character set [0-9a-z-A-z '()+, -./:=?].

2598 – Key Usage (4.2.1.3)

2599 The key usage extension defines the purpose (e.g., encipherment, signature, certificate
2600 signing) of the key contained in the certificate. The usage restriction might be employed
2601 when a key that could be used for more than one operation is to be restricted.

2602 This document does not modify the referenced definition of this extension.

2603 – Basic Constraints (4.2.1.9)
2604 The basic constraints extension identifies whether the subject of the certificate is a CA and
2605 the maximum depth of valid certification paths that include this certificate. Without this
2606 extension, a certificate cannot be an issuer of other certificates.
2607 This document does not modify the referenced definition of this extension.
2608 – Extended Key Usage (4.2.1.12)
2609
2610 Extended Key Usage describes allowed purposes for which the certified public key may can
2611 be used. When a Device receives a certificate, it determines the purpose based on the
2612 context of the interaction in which the certificate is presented, and verifies the certificate can
2613 be used for that purpose.
2614 This document makes the following modifications to the referenced definition of this extension:
2615 CAs SHOULD mark this extension as critical.
2616 CAs MUST NOT issue certificates with the anyExtendedKeyUsage OID (2.5.29.37.0).
2617
2618 The list of OCF-specific purposes and the assigned OIDs to represent them are:
2619 – Identity certificate 1.3.6.1.4.1.44924.1.6
2620 – Role certificate 1.3.6.1.4.1.44924.1.7
2621 **9.4.2.4 Cipher Suite for Authentication, Confidentiality and Integrity**
2622 See 9.4.3.5 for details.
2623 **9.4.2.5 Encoding of Certificate**
2624 See 9.4.2 for details.
2625 **9.4.3 Certificate Revocation List (CRL) Profile**
2626 **9.4.3.1 CRL General**
2627 This clause provides a profile for Certificates Revocation Lists (or CRLs) to facilitate their use
2628 within OCF applications for those communities wishing to support revocation features in their
2629 PKIs.
2630 The OCF CRL profile is derived from IETF RFC 5280 and supports the syntax specified in
2631 IETF RFC 5280 – Clause 5.1
2632 **9.4.3.2 CRL Profile and Fields**
2633 This clause intentionally left empty.
2634 **9.4.3.3 Encoding of CRL**
2635 The ASN.1 distinguished encoding rules (DER method of encoding) defined in [ISO/IEC 8825-1]
2636 should be used to encode CRL.
2637 **9.4.3.4 CRLs Supported Standard Extensions**
2638 The extensions defined by ANSI X9, ISO/IEC, and ITU-T for X.509 v2 CRLs [X.509] [X9.55]
2639 provide methods for associating additional attributes with CRLs. The following list of X.509
2640 extensions should be supported in this certificate profile:
2641 – Authority Key Identifier (Optional; non-critical) - The authority key identifier extension provides
2642 a means of identifying the public key corresponding to the private key used to sign a CRL.
2643 Conforming CRL issuers should use the key identifier method, and shall include this extension
2644 in all CRLs issued

2645 – CRL Number (Optional; non-critical) - The CRL number is a non-critical CRL extension that
2646 conveys a monotonically increasing sequence number for a given CRL scope and CRL issuer

2647 CRL Entry Extensions: The CRL entry extensions defined by ISO/IEC, ITU-T, and ANSI X9 for
2648 X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509]
2649 [X9.55]. Although this document does not provide any recommendation about the use of specific
2650 extensions for CRL entries, conforming CAs may use them in CRLs as long as they are not
2651 marked critical.

2652 **9.4.3.5 Encryption Ciphers and TLS support**

2653 OCF compliant entities shall support TLS version 1.2. Compliant entities shall support
2654 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 cipher suite as defined in IETF RFC 7251 and
2655 may support additional ciphers as defined in the TLS v1.2 specifications.

2656 **9.4.4 Resource Model**

2657 Device certificates and private keys are kept in cred Resource. CRL is maintained and updated
2658 with a separate crl Resource that is defined for maintaining the revocation list.

2659 The cred Resource contains the certificate information pertaining to the Device. The PublicData
2660 Property holds the device certificate and CA certificate chain. PrivateData Property holds the
2661 Device private key paired to the certificate. (See 13.3 for additional detail regarding the
2662 "/oic/sec/cred" Resource).

2663 A certificate revocation list Resource is used to maintain a list of revoked certificates obtained
2664 through the CMS. The Device must consider revoked certificates as part of certificate path
2665 verification. If the CRL Resource is stale or there are insufficient Platform Resources to maintain
2666 a full list, the Device must query the CMS for current revocation status. (See 13.4 for additional
2667 detail regarding the "/oic/sec/crl" Resource).

2668 **9.4.5 Certificate Provisioning**

2669 The CMS (e.g. a hub or a smart phone) issues certificates for new Devices. The CMS shall have
2670 its own certificate and key pair. The certificate is either a) self-signed if it acts as Root CA or b)
2671 signed by the upper CA in its trust hierarchy if it acts as Sub CA. In either case, the certificate
2672 shall have the format described in 9.4.2.

2673 The CA in the CMS shall retrieve a Device's public key and proof of possession of the private key,
2674 generate a Device's certificate signed by this CA certificate, and then the CMS shall transfer
2675 them to the Device including its CA certificate chain. Optionally, the CMS may also transfer one
2676 or more role certificates, which shall have the format described in Clause 9.4.2. . The
2677 subjectPublicKey of each role certificate shall match the subjectPublicKey in the Device
2678 certificate.

2679 In the sequence in Figure 29, the Certificate Signing Request (CSR) is defined by PKCS#10 in
2680 IETF RFC 2986, and is included here by reference.

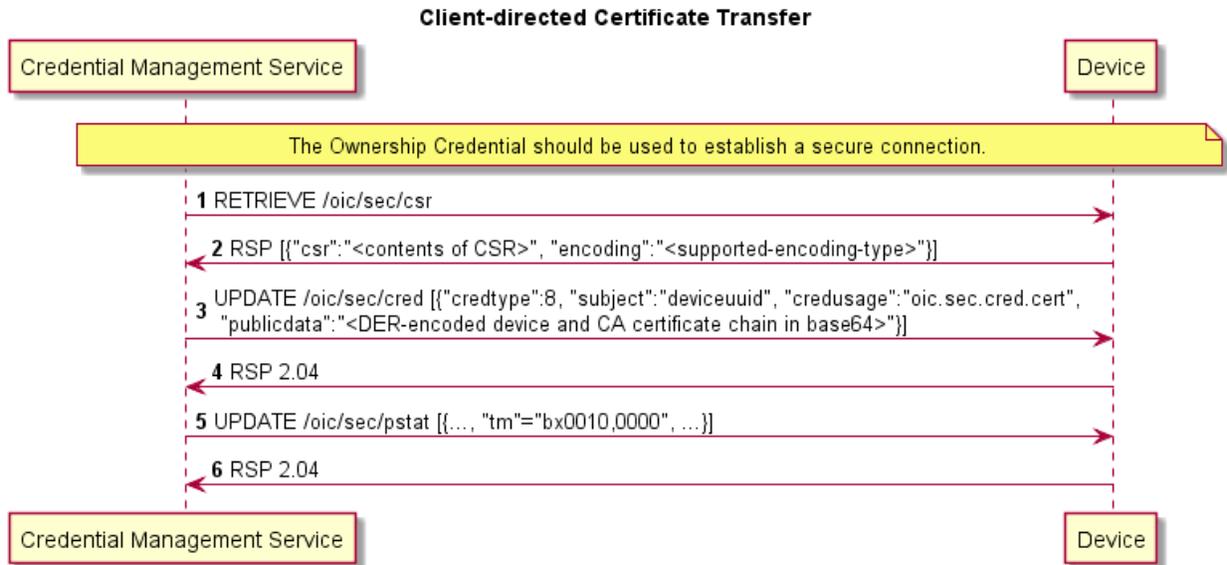
2681 The sequence flow of a certificate transfer for a Client-directed model is described in Figure 29.

2682 1) The CMS retrieves a CSR from the Device that requests a certificate. In this CSR, the Device
2683 shall place its requested UUID into the subject and its public key in the SubjectPublicKeyInfo.
2684 The Device determines the public key to present; this may be an already-provisioned key it
2685 has selected for use with authentication, or if none is present, it may generate a new key pair
2686 internally and provide the public part. The key pair shall be compatible with the allowed
2687 ciphersuites listed in 9.4.2.4 and 11.3.4, since the certificate will be restricted for use in OCF
2688 authentication.

2689 2) If the Device does not have a pre-provisioned key pair and is unable to generate a key pair on
2690 its own, then it is not capable of using certificates. The Device shall advertise this fact both by

2691 setting the 0x8 bit position in the sct Property of /oic/sec/doxm to 0, and return an error that
 2692 the /oic/sec/csr resource does not exist.

2693 3) The CMS shall transfer the issued certificate and CA chain to the designated Device using
 2694 the same credid, to maintain the association with the private key. The credential type
 2695 (oic.sec.cred) used to transfer certificates in Figure 29 is also used to transfer role certificates,
 2696 by including multiple credentials in the POST from CMS to Device. Identity certificates shall
 2697 be stored with the credusage Property set to `oic.sec.cred.cert` and role certificates shall be
 2698 stored with the credusage Property set to `oic.sec.cred.rolecert`.



2699

2700 **Figure 29 – Client-directed Certificate Transfer**

2701 **9.4.6 CRL Provisioning**

2702 The only pre-requirement of CRL issuing is that CMS (e.g. a hub or a smart phone) has the
 2703 function to register revocation certificates, to sign CRL and to transfer it to Devices.

2704 The CMS sends the CRL to the Device.

2705 Any certificate revocation reasons listed below cause CRL update on each Device.

- 2706 – change of issuer name
- 2707 – change of association between Devices and CA
- 2708 – certificate compromise
- 2709 – suspected compromise of the corresponding private key

2710 CRL may be updated and delivered to all accessible Devices in the OCF Security Domain. In
 2711 some special cases, Devices may request CRL to a given CMS.

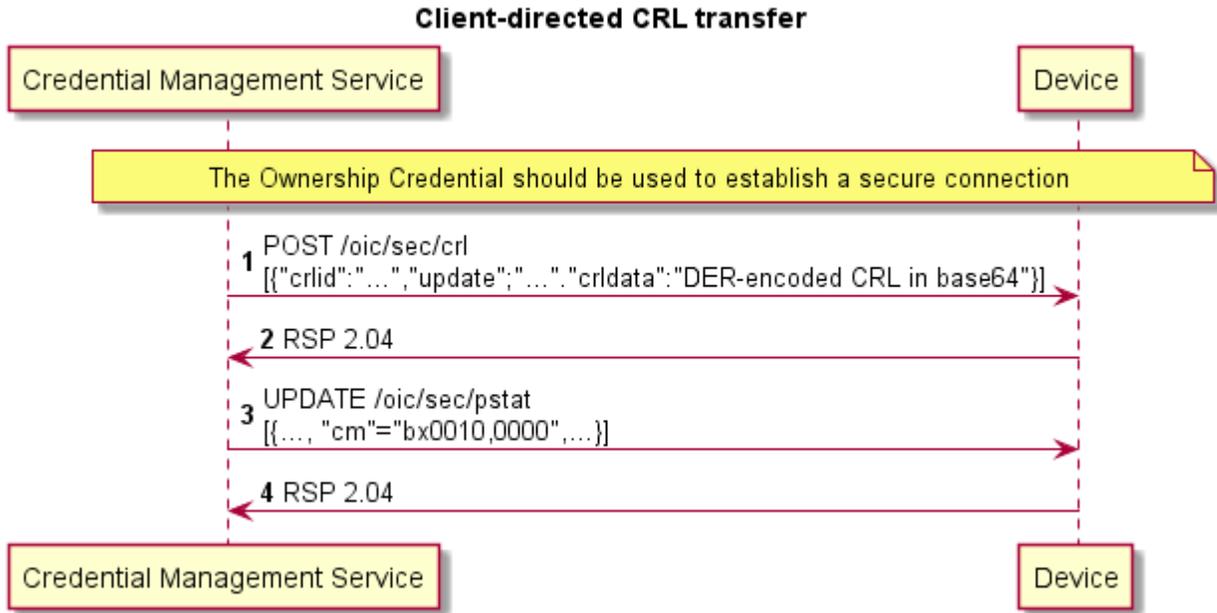
2712 There are two options to update and deliver CRL;

- 2713 – CMS pushes CRL to each Device
- 2714 – each Device periodically requests to update CRL

2715 The sequence flow of a CRL transfer for a Client-directed model is described in Figure 30.

2716 1) The CMS may retrieve the CRL Resource Property.

2717 2) If the Device requests the CMS to send CRL, it should transfer the latest CRL to the Device.
 2718
 2719



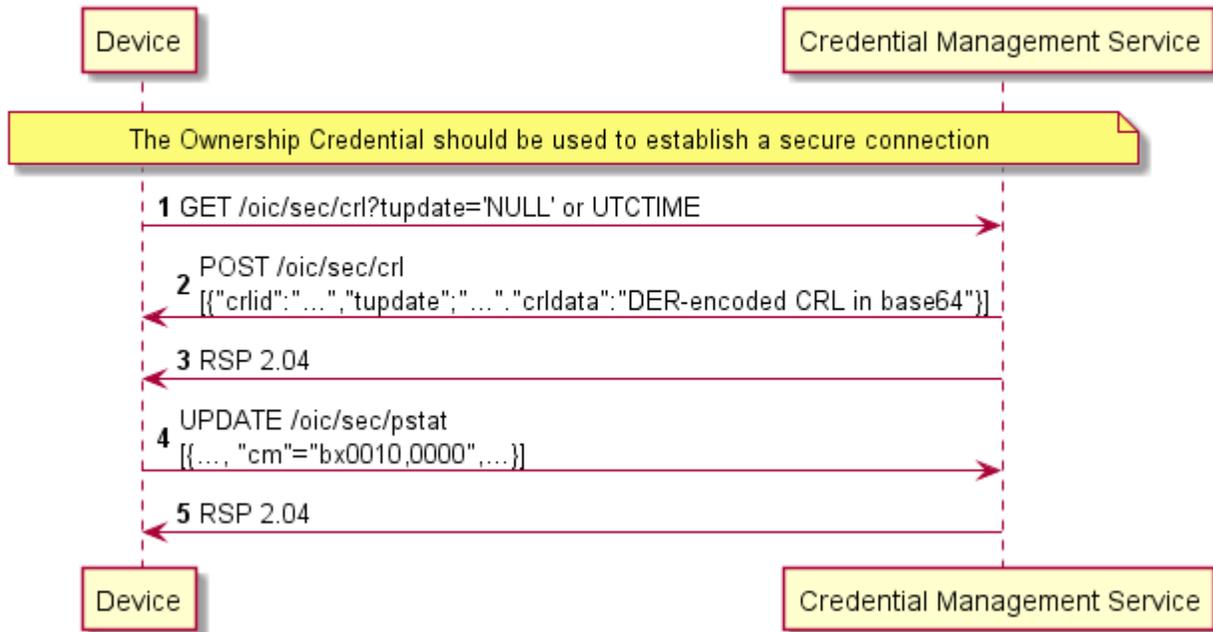
2720

Figure 30 – Client-directed CRL Transfer

2721 The sequence flow of a CRL transfer for a Server-directed model is described in Figure 31.

- 2722 1) The Device retrieves the CRL Resource Property tupdate to the CMS.
 2723 2) If the CMS recognizes the updated CRL information after the designated tupdate time, it may
 2724 transfer its CRL to the Device.

Server-directed CRL transfer



2725
2726
2727

Figure 31 – Server-directed CRL Transfer

2728 **10 Device Authentication**

2729 **10.1 Device Authentication General**

2730 When a Client is accessing a restricted Resource on a Server, the Server shall authenticate the
2731 Client. Clients shall authenticate Servers while requesting access. Clients may also assert one or
2732 more roles that the server can use in access control decisions. Roles may be asserted when the
2733 Device authentication is done with certificates.

2734 **10.2 Device Authentication with Symmetric Key Credentials**

2735 When using symmetric keys to authenticate, the Server Device shall include the
2736 ServerKeyExchange message and set `psk_identity_hint` to the Server's Device ID. The Client
2737 shall validate that it has a credential with the Subject ID set to the Server's Device ID, and a
2738 credential type of PSK. If it does not, the Client shall respond with an `unknown_psk_identity` error
2739 or other suitable error.

2740 If the Client finds a suitable PSK credential, it shall reply with a ClientKeyExchange message that
2741 includes a `psk_identity_hint` set to the Client's Device ID. The Server shall verify that it has a
2742 credential with the matching Subject ID and type. If it does not, the Server shall respond with an
2743 `unknown_psk_identity` or other suitable error code. If it does, then it shall continue with the DTLS
2744 protocol, and both Client and Server shall compute the resulting premaster secret.

2745 **10.3 Device Authentication with Raw Asymmetric Key Credentials**

2746 When using raw asymmetric keys to authenticate, the Client and the Server shall include a
2747 suitable public key from a credential that is bound to their Device. Each Device shall verify that
2748 the provided public key matches the `PublicData` field of a credential they have, and use the
2749 corresponding Subject ID of the credential to identify the peer Device.

2750 **10.4 Device Authentication with Certificates**

2751 **10.4.1 Device Authentication with Certificates General**

2752 When using certificates to authenticate, the Client and Server shall each include their certificate
2753 chain, as stored in the appropriate credential, as part of the selected authentication cipher suite.
2754 Each Device shall validate the certificate chain presented by the peer Device. Each certificate
2755 signature shall be verified until a public key is found within the `/oic/sec/cred` Resource with the
2756 `'oic.sec.cred.trustca'` credusage. Credential Resource found in `/oic/sec/cred` are used to
2757 terminate certificate path validation. Also, the validity period and revocation status should be
2758 checked for all above certificates, but at this time a failure to obtain a certificate's revocation
2759 status (CRL or OCSP response) MAY continue to allow the use of the certificate if all other
2760 verification checks succeed.

2761 If available, revocation information should be used to verify the revocation status of the certificate.
2762 The URL referencing the revocation information should be retrieved from the certificate (via the
2763 `authorityInformationAccess` or `crlDistributionPoints` extensions). Other mechanisms may be used
2764 to gather relevant revocation information like CRLs or OCSP responses.

2765 Each Device shall use the corresponding Subject ID of the credential to identify the peer Device.

2766 Devices must follow the certificate path validation algorithm in Clause 6 of IETF RFC 5280. In
2767 particular:

- 2768 – For all non-End-Entity certificates, Devices shall verify that the basic constraints extension is
2769 present, and that the `cA` boolean in the extension is `TRUE`. If either is false, the certificate
2770 chain **MUST** be rejected. If the `pathLenConstraint` field is present, Devices will confirm the
2771 number of certificates between this certificate and the End-Entity certificate is less than or
2772 equal to `pathLenConstraint`. In particular, if `pathLenConstraint` is zero, only an End-Entity

2773 certificate can be issued by this certificate. If the pathLenConstraint field is absent, there is no
2774 limit to the chain length.

2775 – For all non-End-Entity certificates, Devices shall verify that the key usage extension is
2776 present, and that the keyCertSign bit is asserted.

2777 – Devices may use the Authority Key Identifier extension to quickly locate the issuing certificate.
2778 Devices MUST NOT reject a certificate for lacking this extension, and must instead attempt
2779 validation with the public keys of possible issuer certificates whose subject name equals the
2780 issuer name of this certificate.

2781 – The End-Entity certificate of the chain shall be verified to contain an Extended Key Usage
2782 (EKU) suitable to the purpose for which it is being presented. An End-Entity certificate which
2783 contains no ECU extension is not valid for any purpose and must be rejected. Any certificate
2784 which contains the anyExtendedKeyUsage OID (2.5.29.37.0) must be rejected, even if other
2785 valid EKUs are also present.

2786 – Devices MUST verify "transitive ECU" for certificate chains. Issuer certificates (any certificate
2787 that is not an End-Entity) in the chain MUST all be valid for the purpose for which the
2788 certificate chain is being presented. An issuer certificate is valid for a purpose if it contains an
2789 ECU extension and the ECU OID for that purpose is listed in the extension, OR it does not
2790 have an ECU extension. An issuer certificate SHOULD contain an ECU extension and a
2791 complete list of EKUs for the purposes for which it is authorized to issue certificates. An
2792 issuer certificate without an ECU extension is valid for all purposes; this differs from End-
2793 Entity certificates without an ECU extension.

2794 The list of purposes and their associated OIDs are defined in 9.4.2.3.

2795 If the Device does not recognize an extension, it must examine the `critical` field. If the field is
2796 TRUE, the Device MUST reject the certificate. If the field is FALSE, the Device MUST treat the
2797 certificate as if the extension were absent and proceed accordingly. This applies to all certificates
2798 in a chain.

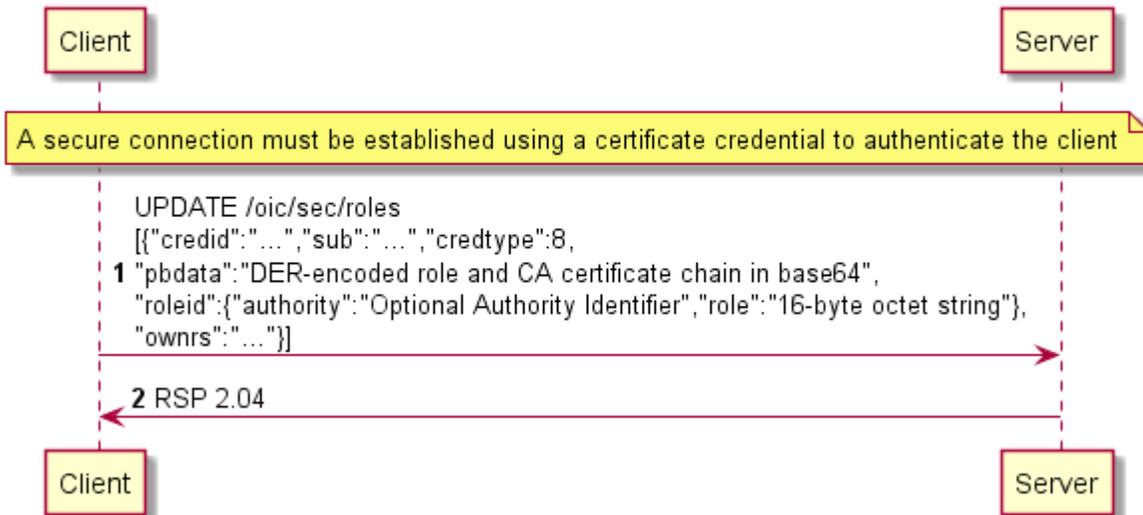
2799 NOTE Certificate revocation mechanisms are currently out of scope of this version of the document.

2800 **10.4.2 Role Assertion with Certificates**

2801 This clause describes role assertion by a client to a server using a certificate role credential. If a
2802 server does not support the certificate credential type, clients should not attempt to assert roles
2803 with certificates.

2804 Following authentication with a certificate, a client may assert one or more roles by updating the
2805 server's roles resource with the role certificates it wants to use. The role credentials must be
2806 certificate credentials and shall include a certificate chain. The server shall validate each
2807 certificate chain as specified in Clause 10.3. Additionally, the public key in the End-Entity
2808 certificate used for Device authentication must be identical to the public key in all role (End-Entity)
2809 certificates. Also, the subject distinguished name in the End-Entity authentication and role
2810 certificates must match. The roles asserted are encoded in the subjectAltName extension in the
2811 certificate. The subjectAltName field can have multiple values, allowing a single certificate to
2812 encode multiple roles that apply to the client. The server shall also check that the ECU extension
2813 of the role certificate(s) contains the value 1.3.6.1.4.1.44924.1.7 (see Clause 9.4.2.2) indicating
2814 the certificate may be used to assert roles. Figure 32 describes how a client Device asserts roles
2815 to a server.

Asserting Certificate Role Credentials



2816

2817

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

2818 Additional comments for Figure 32

2819 1) The response shall contain "204 No Content" to indicate success or 4xx to indicate an error. If
2820 the server does not support certificate credentials, it should return "501 Not Implemented"

2821 2) Roles asserted by the client may be kept for a duration chosen by the server. The duration
2822 shall not exceed the validity period of the role certificate. When fresh CRL information is
2823 obtained, the certificates in "/oic/sec/roles" should be checked, and the role removed if the
2824 certificate is revoked or expired.

2825 3) Servers should choose a nonzero duration to avoid the cost of frequent re-assertion of a role
2826 by a client. It is recommended that servers use the validity period of the certificate as a
2827 duration, effectively allowing the CMS to decide the duration.

2828 4) The format of the data sent in the create call shall be a list of credentials (oic.sec.cred, see
2829 Table 28). They shall have credtype 8 (indicating certificates) and PrivateData field shall not
2830 be present. For fields that are duplicated in the oic.sec.cred object and the certificate, the
2831 value in the certificate shall be used for validation. For example, if the Period field is set in
2832 the credential, the server must treat the validity period in the certificate as authoritative.
2833 Similar for the roleid data (authority, role).

2834 5) Certificates shall be encoded as in Figure 29 (DER-encoded certificate chain in base64)

2835 6) Clients may GET the /oic/sec/roles resource to determine the roles that have been previously
2836 asserted. An array of credential objects shall be returned. If there are no valid certificates
2837 corresponding to the currently connected and authenticated Client's identity, then an empty
2838 array (i.e. []) shall be returned.

2839 10.4.3 OCF PKI Roots

2840 This clause intentionally left empty.

2841 10.4.4 PKI Trust Store

2842 Each Device using a certificate chained to an OCF Root CA trust anchor SHALL securely store
2843 the OCF Root CA certificates in the oic/sec/cred resource and SHOULD physically store this
2844 resource in a hardened memory location where the certificates cannot be tampered with.

2845 **10.4.5 Path Validation and extension processing**

2846 Devices SHALL follow the certificate path validation algorithm in Clause 6 of IETF RFC 5280. In
2847 addition, the following are best practices and SHALL be adhered to by any OCF-compliant
2848 application handling digital certificates

2849 – Validity Period checking

2850 OCF-compliant applications SHALL conform to IETF RFC 5280 clauses 4.1.2.5, 4.1.2.5.1, and
2851 4.1.2.5.2 when processing the notBefore and notAfter fields in X.509 certificates. In addition,
2852 for all certificates, the notAfter value SHALL NOT exceed the notAfter value of the issuing CA.

2853 – Revocation checking

2854 Relying applications SHOULD check the revocation status for all certificates, but at this time,
2855 an application MAY continue to allow the use of the certificate upon a failure to obtain a
2856 certificate's revocation status (CRL or OCSP response), if all other verification checks
2857 succeed.

2858 – basicConstraints

2859 For all Root and Intermediate Certificate Authority (CA) certificates, Devices SHALL verify
2860 that the basicConstraints extension is present, flagged critical, and that the cA boolean value
2861 in the extension is TRUE. If any of these are false, the certificate chain SHALL be rejected.

2862 If the pathLenConstraint field is present, Devices will confirm the number of certificates
2863 between this certificate and the End-Entity certificate is less than or equal to
2864 pathLenConstraint. In particular, if pathLenConstraint is zero, only an End-Entity certificate
2865 can be issued by this certificate. If the pathLenConstraint field is absent, there is no limit to
2866 the chain length.

2867 For End-Entity certificates, if the basicConstraints extension is present, it SHALL be flagged
2868 critical, SHALL have a cA boolean value of FALSE, and SHALL NOT contain a
2869 pathLenConstraint ASN.1 sequence. An End-Entity certificate SHALL be rejected if a
2870 pathLenConstraint ASN.1 sequence is either present with an Integer value, or present with a
2871 null value.

2872 In order to facilitate future flexibility in OCF-compliant PKI implementations, all OCF-
2873 compliant Root CA certificates SHALL NOT contain a pathLenConstraint. This allows
2874 additional tiers of Intermediate CAs to be implemented in the future without changing the Root
2875 CA trust anchors, should such a requirement emerge.

2876 – keyUsage

2877 For all certificates, Devices shall verify that the key usage extension is present and flagged
2878 critical.

2879 For Root and Intermediate CA certificates, ONLY the keyCertSign(5) and crlSign(6) bits
2880 SHALL be asserted.

2881 For End-Entity certificates, ONLY the digitalSignature(0) and keyAgreement(4) bits SHALL be
2882 asserted.

2883 – extendedKeyUsage:

2884 Any End-Entity certificate containing the anyExtendedKeyUsage OID (2.5.29.37.0) SHALL be
2885 rejected.

2886 OIDs for serverAuthentication (1.3.6.1.5.5.7.3.1) and clientAuthentication (1.3.6.1.5.5.7.3.2)
2887 are required for compatibility with various TLS implementations.

2888 At this time, an End-Entity certificate cannot be used for both Identity (1.3.6.1.4.1.44924.1.6)
2889 and Role (1.3.6.1.4.1.44924.1.7) purposes. Therefore, exactly one of the two OIDs SHALL be
2890 present and End-Entity certificates with EKU extensions containing both OIDs SHALL be
2891 rejected.

2892 – certificatePolicies

2893 End-Entity certificates which chain to an OCF Root CA SHOULD contain at least one
2894 PolicyIdentifierId set to the OCF Certificate Policy OID – (1.3.6.1.4.1.51414.0.1.2)
2895 corresponding to the version of the OCF Certificate Policy under which it was issued.
2896 Additional manufacturer-specific CP OIDs may also be populated.

2897 **10.5 Device Authentication with OCF Cloud**

2898 **10.5.1 Device Authentication with OCF Cloud General**

2899 The mechanisms for Device Authentication in clauses 10.2, 10.3 and 10.4 imply that a Device is
2900 authorized to communicate with any other Device meeting the criteria provisioned in /oic/sec/cred;
2901 the “/oic/sec/acl2” Resource (or “/oic/sec/acl1” resource of OIC1.1 Servers) are additionally used
2902 to restrict access to specific Resources. The present clause describes Device authentication for
2903 OCF Cloud, which uses slightly different criteria as described in clause 5. A Device accessing an
2904 OCF Cloud shall establish a TLS session. The mutual authenticated TLS session is established
2905 using Server certificate and Client certificate.

2906 Each Device is identified based on the Access Token it is assigned during Device Registration.
2907 The OCF Cloud holds an OCF Cloud association table that maps Access Token, User ID and
2908 Device ID. The Device Registration shall happen while the Device is in RFNOP state. After
2909 Device Registration, the updated Access Token, Device ID and User ID are used by the Device
2910 for the subsequent connection with the OCF Cloud.

2911 **10.5.2 Device Connection with the OCF Cloud**

2912 The Device should establish the TLS connection using the certificate based credential. The
2913 connection should be established after Device is provisioned by Mediator.

2914 The TLS session is established between Device and the OCF Cloud as specified in IETF RFC
2915 8323. The OCF Cloud is expected to provide certificate signed by trust anchor that is present in
2916 cred entries of the Device. These cred entries are expected to be configured by the Mediator.

2917 The Device shall validate the OCF Cloud's identity based on the credentials that are contained in
2918 /oic/sec/cred Resource entries of the Device.

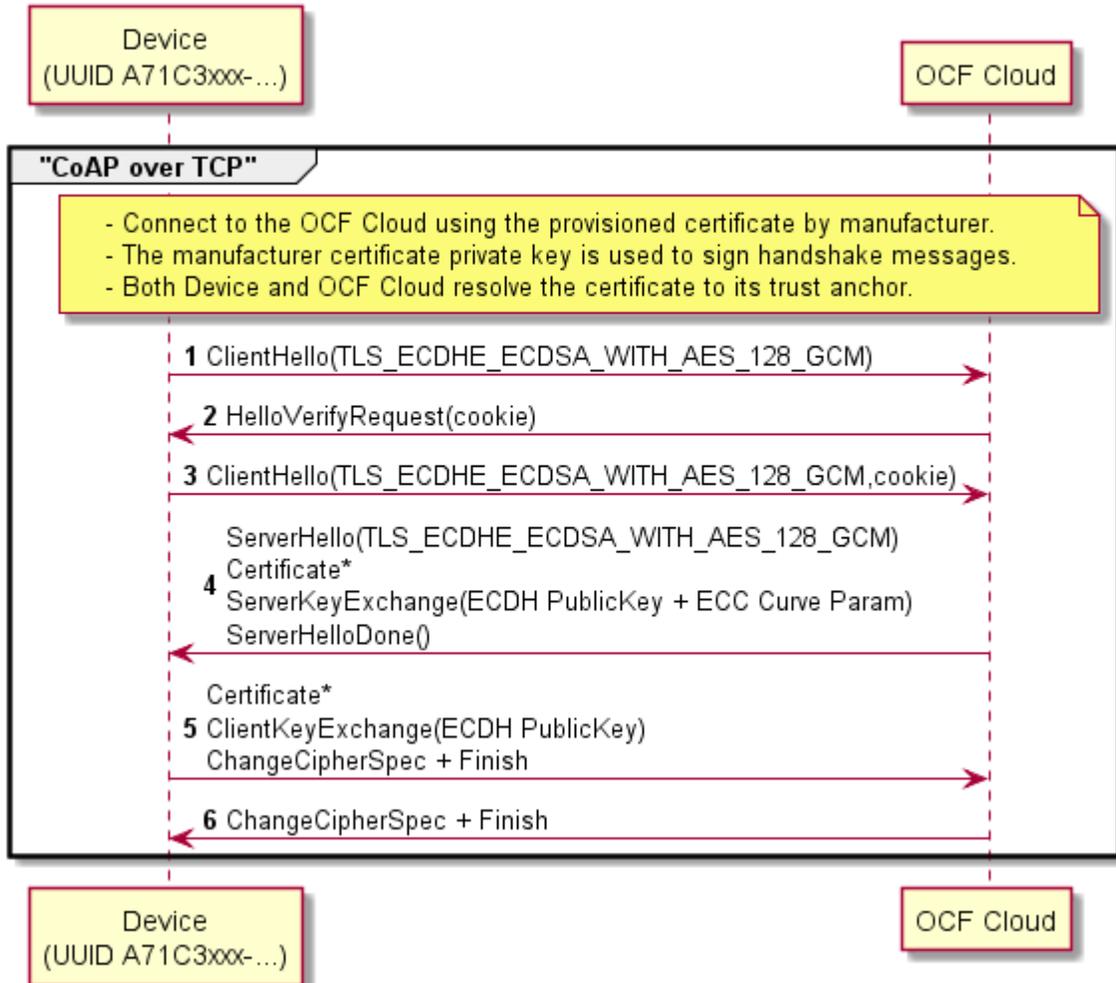
2919 The OCF Cloud is expected to validate the manufacturer certificate provided by the Device.

2920 The assumption is that the OCF Cloud User trusts the OCF Cloud that the Device connects. The
2921 OCF Cloud connection should not happen without the consent of the OCF Cloud User. The
2922 assumption is that the OCF Cloud User has either service agreement with the OCF Cloud
2923 provider or uses manufacturer provided OCF Cloud.

2924 If authentication fails, the “clec” Property of oic.r.coapcloudconf Resource on the Device shall be
2925 updated about the failed state, if it is supported by the Device. If authentication succeeds, the
2926 Device and OCF Cloud should establish an encrypted link in accordance with the negotiated
2927 cipher suite.

2928 Figure 33 depicts sequence for Device connection with OCF Cloud and steps described in
2929 Table 22.

Device Connection with OCF Cloud



2930

2931

Figure 33 – Device connection with OCF Cloud

2932

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

2933

10.5.3 Security Considerations

2934

2935

2936

2937

2938

2939

2940

2941

2942

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

2943 The security considerations about the manufacturer certificate as described in 7.3.6.5 are also
2944 applicable in the Device authentication with the OCF Cloud.

2945 The Device should validate the OCF Cloud's TLS certificate as defined by IETF RFC 6125 and in
2946 accordance with its requirements for Server identity authentication.

2947 The "uid" and "di" Property Value of "/oic/d" Resource may be considered personally identifiable
2948 information in some regulatory regions, and the OCF Cloud is expected to provide protections
2949 appropriate to its governing regulatory bodies.

2950

2951 **11 Message Integrity and Confidentiality**

2952 **11.1 Preamble**

2953 Secured communications between Clients and Servers are protected against eavesdropping,
2954 tampering, or message replay, using security mechanisms that provide message confidentiality
2955 and integrity.

2956 **11.2 Session Protection with DTLS**

2957 **11.2.1 DTLS Protection General**

2958 Devices shall support DTLS for secured communications as defined in IETF RFC 6347. Devices
2959 using TCP shall support TLS v1.2 for secured communications as defined in IETF RFC 5246. See
2960 11.3 for a list of required and optional cipher suites for message communication.

2961 OCF Devices MUST support (D)TLS version 1.2 or greater and MUST NOT support versions 1.1
2962 or lower.

2963 Multicast session semantics are not yet defined in this version of the security document.

2964 **11.2.2 Unicast Session Semantics**

2965 For unicast messages between a Client and a Server, both Devices shall authenticate each other.
2966 See Clause 10 for details on Device Authentication.

2967 Secured unicast messages between a Client and a Server shall employ a cipher suite from 11.3.
2968 The sending Device shall encrypt and authenticate messages as defined by the selected cipher
2969 suite and the receiving Device shall verify and decrypt the messages before processing them.

2970 **11.2.3 Cloud Session Semantics**

2971 The messages between the OCF Cloud and Device shall be exchanged only if the Device and
2972 OCF Cloud authenticate each other as described in 10.4.3. The asymmetric cipher suites as
2973 described in 11.3.5 shall be employed for establishing a secured session and for
2974 encrypting/decrypting between the OCF Cloud and the Device. The OCF Endpoint sending the
2975 message shall encrypt and authenticate the message using the cipher suite as described in
2976 11.3.5 and the OCF Endpoint shall verify and decrypt the message before processing it.

2977 **11.3 Cipher Suites**

2978 **11.3.1 Cipher Suites General**

2979 The cipher suites allowed for use can vary depending on the context. This clause lists the cipher
2980 suites allowed during ownership transfer and normal operation. The following RFCs provide
2981 additional information about the cipher suites used in OCF.

2982 IETF RFC 4279: Specifies use of pre-shared keys (PSK) in (D)TLS

2983 IETF RFC 4492: Specifies use of elliptic curve cryptography in (D)TLS

2984 IETF RFC 5489: Specifies use of cipher suites that use elliptic curve Diffie-Hellman (ECDHE) and
2985 PSKs

2986 IETF RFC 6655 and IETF RFC 7251: Specifies AES-CCM mode cipher suites, with ECDHE

2987 **11.3.2 Cipher Suites for Device Ownership Transfer**

2988 **11.3.2.1 Just Works Method Cipher Suites**

2989 The Just Works OTM may use the following (D)TLS cipher suites.

2990 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256,

2991 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256

2992 All Devices supporting Just Works OTM shall implement:

2993 TLS_ECDH_ANON_WITH_AES_128_CBC_SHA256 (with the value 0xFF00)

2994 All Devices supporting Just Works OTM should implement:

2995 TLS_ECDH_ANON_WITH_AES_256_CBC_SHA256 (with the value 0xFF01)

2996 **11.3.2.2 Random PIN Method Cipher Suites**

2997 The Random PIN Based OTM may use the following (D)TLS cipher suites.

2998 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

2999 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

3000 All Devices supporting Random Pin Based OTM shall implement:

3001 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256

3002 **11.3.2.3 Certificate Method Cipher Suites**

3003 The Manufacturer Certificate Based OTM may use the following (D)TLS cipher suites.

3004 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

3005 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

3006 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

3007 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

3008 Using the following curve:

3009 secp256r1 (See IETF RFC 4492)

3010 All Devices supporting Manufacturer Certificate Based OTM shall implement:

3011 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

3012 Devices supporting Manufacturer Certificate Based OTM should implement:

3013 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,

3014 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,

3015 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

3016 **11.3.3 Cipher Suites for Symmetric Keys**

3017 The following cipher suites are defined for (D)TLS communication using PSKs:

3018 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,

3019 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,

3020 TLS_PSK_WITH_AES_128_CCM_8, (* 8 OCTET Authentication tag *)

3021 TLS_PSK_WITH_AES_256_CCM_8,

3022 TLS_PSK_WITH_AES_128_CCM, (* 16 OCTET Authentication tag *)

3023 TLS_PSK_WITH_AES_256_CCM,

3024 All CCM based cipher suites also use HMAC-SHA-256 for authentication.

3025 All Devices shall implement the following:

3026 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
3027

3028 Devices should implement the following:

3029 TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,
3030 TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA256,
3031 TLS_PSK_WITH_AES_128_CCM_8,
3032 TLS_PSK_WITH_AES_256_CCM_8,
3033 TLS_PSK_WITH_AES_128_CCM,
3034 TLS_PSK_WITH_AES_256_CCM

3035 **11.3.4 Cipher Suites for Asymmetric Credentials**

3036 The following cipher suites are defined for (D)TLS communication with asymmetric keys or
3037 certificates:

3038 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,
3039 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
3040 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
3041 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

3042 Using the following curve:

3043 secp256r1 (See IETF RFC 4492)

3044 All Devices supporting Asymmetric Credentials shall implement:

3045 TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

3046 All Devices supporting Asymmetric Credentials should implement:

3047 TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8,
3048 TLS_ECDHE_ECDSA_WITH_AES_128_CCM,
3049 TLS_ECDHE_ECDSA_WITH_AES_256_CCM

3050 **11.3.5 Cipher suites for OCF Cloud Credentials**

3051 The following cipher suites are defined for TLS communication with certificates:

3052 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
3053 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
3054 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
3055 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
3056 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
3057 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

3058 All Devices supporting OCF Cloud Certificate Credentials shall implement:

3059 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

3060 All Devices supporting OCF Cloud Certificate Credentials should implement:

3061 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
3062 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,

3063 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
3064 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
3065 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
3066

3067 **12 Access Control**

3068 **12.1 ACL Generation and Management**

3069 This clause will be expanded in a future version of the document.

3070 **12.2 ACL Evaluation and Enforcement**

3071 **12.2.1 ACL Evaluation and Enforcement General**

3072 The Server enforces access control over application Resources before exposing them to the
3073 requestor. The Security Layer in the Server authenticates the requestor when access is received
3074 via the secure port. Authenticated requestors, known as the "subject" can be used to match ACL
3075 entries that specify the requestor's identity, role or may match authenticated requestors using a
3076 subject wildcard.

3077 If the request arrives over the unsecured port, the only ACL policies allowed are those that use a
3078 subject wildcard match of anonymous requestors.

3079 Access is denied if a requested Resource is not matched by an ACL entry.

3080 NOTE There are documented exceptions pertaining to Device onboarding where access to Security Virtual Resources
3081 may be granted prior to provisioning of ACL Resources.

3082 The second generation ACL (i.e. "/oic/sec/acl2") contains an array of Access Control Entries
3083 (ACE2) that employ a Resource matching algorithm that uses an array of Resource references to
3084 match Resources to which the ACE2 access policy applies. Matching consists of comparing the
3085 values of the ACE2 "resources" Property (see Clause 13) to the requested Resource. Resources
3086 are matched in two ways:

- 3087 1) host reference (href)
- 3088 2) resource wildcard (wc).

3089 **12.2.2 Host Reference Matching**

3090 When present in an ACE2 matching element, the Host Reference (href) Property shall be used for
3091 Resource matching.

3092 – The href Property shall be used to find an exact match of the Resource name if present.

3093 **12.2.3 Resource Wildcard Matching**

3094 When present, a wildcard (wc) expression shall be used to match multiple Resources using a
3095 wildcard Property contained in the oic.sec.ace2.resource-ref structure.

3096 A wildcard expression may be used to match multiple Resources using a wildcard Property
3097 contained in the oic.sec.ace2.resource-ref structure. The wildcard matching strings are defined in
3098 Table 23.

3099 **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.

3100 NOTE Discoverable resources appear in the "/oic/wk/res" Resource, while non-discoverable resources may appear in
3101 other collection resources but do not appear in the /res collection.

3102 **12.2.4 Multiple Criteria Matching**

3103 If the ACE2 “resources” Property contains multiple entries, then a logical OR shall be applied for
3104 each array element. For example, if a first array element of the “resources” Property contains
3105 ‘href’=“/a/light” and the second array element of the “resources” Property contains ‘href’=“/a/led”,
3106 then Resources that match either of the two ‘href’ criteria shall be included in the set of matched
3107 Resources.

3108 Example 1 JSON for Resource matching

```
3109 {  
3110 //Matches Resources named "/x/door1" or "/x/door2"  
3111 "resources": [  
3112   {  
3113     "href": "/x/door1"  
3114   },  
3115   {  
3116     "href": "/x/door2"  
3117   },  
3118 ]  
3119 }
```

3120 Example 2 JSON for Resource matching

```
3121 {  
3122 // Matches all Resources  
3123 "resources": [  
3124   {  
3125     "wc": "*"   
3126   }  
3127 ]  
3128 }
```

3129 **12.2.5 Subject Matching using Wildcards**

3130 When the ACE subject is specified as the wildcard string "*" any requestor is matched. The OCF
3131 server may authenticate the OCF client, but is not required to.

3132 Examples: JSON for subject wildcard matching

```
3133 //matches all subjects that have authenticated and confidentiality protections in place.  
3134 "subject" : {  
3135   "conntype" : "auth-crypt"  
3136 }  
3137 //matches all subjects that have NOT authenticated and have NO confidentiality protections in place.  
3138 "subject" : {  
3139   "conntype" : "anon-clear"  
3140 }
```

3141 **12.2.6 Subject Matching using Roles**

3142 When the ACE subject is specified as a role, a requestor shall be matched if either:

- 3143 1) The requestor authenticated with a symmetric key credential, and the role is present in the
3144 roleid Property of the credential’s entry in the credential resource, or

3145 2) The requestor authenticated with a certificate, and a valid role certificate is present in the
3146 roles resource with the requestor's certificate's public key at the time of evaluation. Validating
3147 role certificates is defined in 10.3.1.

3148 **12.2.7 ACL Evaluation**

3149 **12.2.7.1 ACE2 matching algorithm**

3150 The OCF Server shall apply an ACE2 matching algorithm that matches in the following sequence:

- 3151 1) If the "/oic/sec/sacl" Resource exists and if the signature verification is successful, these
3152 ACE2 entries contribute to the set of local ACE2 entries in step 3. The Server shall verify the
3153 signature, at least once, following update of the "/oic/sec/sacl" Resource.
- 3154 2) The local /oic/sec/acl2 Resource contributes its ACE2 entries for matching.
- 3155 3) Access shall be granted when all these criteria are met:
 - 3156 a) The requestor is matched by the ACE2 "subject" Property.
 - 3157 b) The requested Resource is matched by the ACE2 resources PropertyProperty and the
3158 requested Resource shall exist on the local Server.
 - 3159 c) The "period" Property constraint shall be satisfied.
 - 3160 d) The "permission" Property constraint shall be applied.

3161 If multiple ACE2 entries match the Resource request, the union of permissions, for all matching
3162 ACEs, defines the *effective* permission granted. E.g. If Perm1=CR---; Perm2=--UDN; Then
3163 UNION (Perm1, Perm2)=CRUDN.

3164 The Server shall enforce access based on the effective permissions granted.

3165 Batch requests to Resource containing Links require additional considerations when accessing
3166 the linked Resources. ACL considerations for batch request to the Atomic Measurement
3167 Resource Type are provided in clause 12.2.7.2. ACL considerations for batch request to the
3168 Collection Resource Type are provided in 12.2.7.3.

3169 **12.2.7.2 ACL considerations for batch request to the Atomic Measurement Resource 3170 Type**

3171 The present clause shall apply to any Resource Type based on the Atomic Measurement
3172 Resource Type.

3173 If an OCF Server receives a batch request to an Atomic Measurement Resource containing only
3174 local references and there is an ACE matching the Atomic Measurement Resource which permits
3175 the request, then the corresponding requests to the linked Resources of the Atomic Measurement
3176 Resource shall be permitted by the OCF Server. That is, the request to each linked Resource is
3177 permitted regardless of whether there is an ACE configured on the OCF Server which would
3178 permit a corresponding request from the OCF Client (which sent the batch request to the Atomic
3179 Measurement Resource) addressing the linked Resource.

3180 **12.2.7.3 ACL considerations for batch request to the Collection Resource Type**

3181 The present clause shall apply to any Resource Type based on the Collection Resource Type.

3182 If an OCF Server receives a batch request to a Collection Resource containing only local
3183 references and there is an ACE matching the Collection Resource which permits the request,
3184 then the corresponding requests to the linked Resources of the Collection Resource shall be
3185 permitted by the OCF Server. That is, the request to each linked Resource is permitted
3186 regardless of whether there is an ACE configured on the OCF Server which would permit a
3187 corresponding request from the OCF Client (which sent the batch request to the Collection
3188 Resource) addressing the linked Resource.

3190 **13 Security Resources**

3191 **13.1 Security Resources General**

3192 OCF Security Resources are shown in Figure 34.

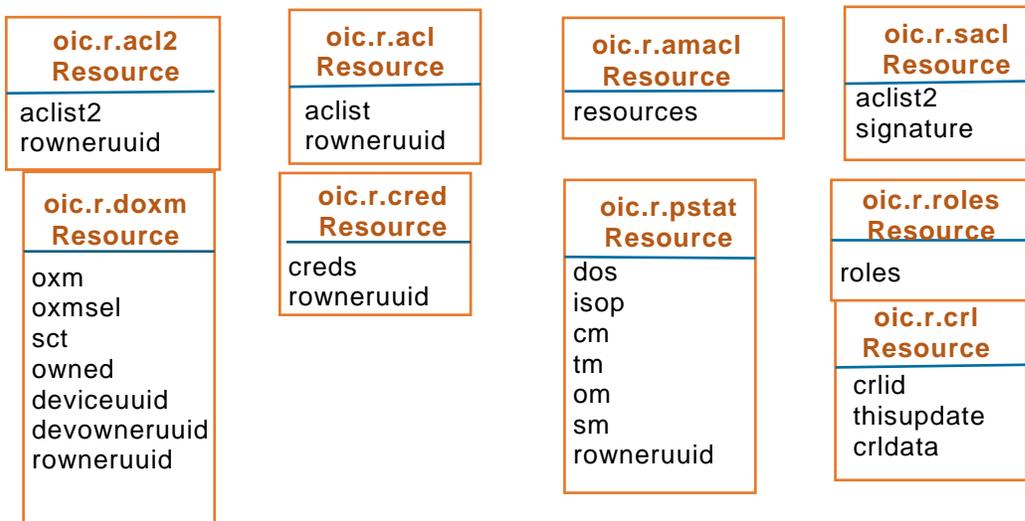
3193 "/oic/sec/cred" Resource and Properties are shown in Figure 35.

3194 "/oic/sec/acl2" Resource and Properties are shown in Figure 36.

3195 "/oic/sec/amacl" Resource and Properties are shown in Figure 37.

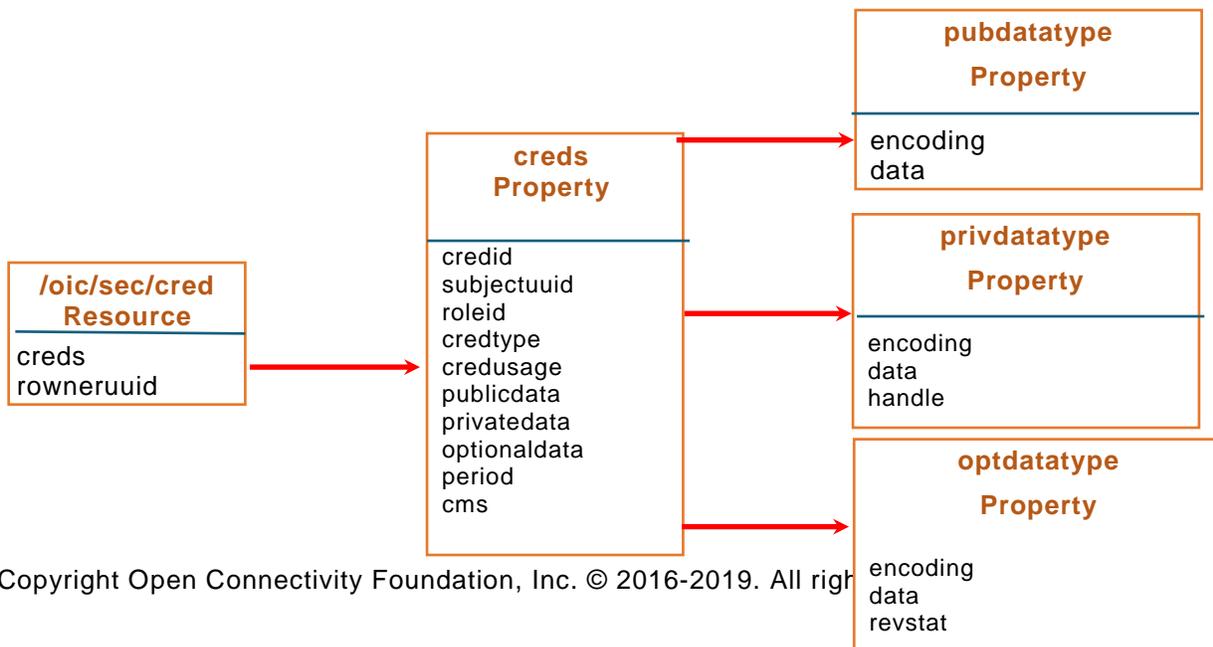
3196 "/oic/sec/sacl" Resource and Properties are shown in Figure 38.

3197



3198

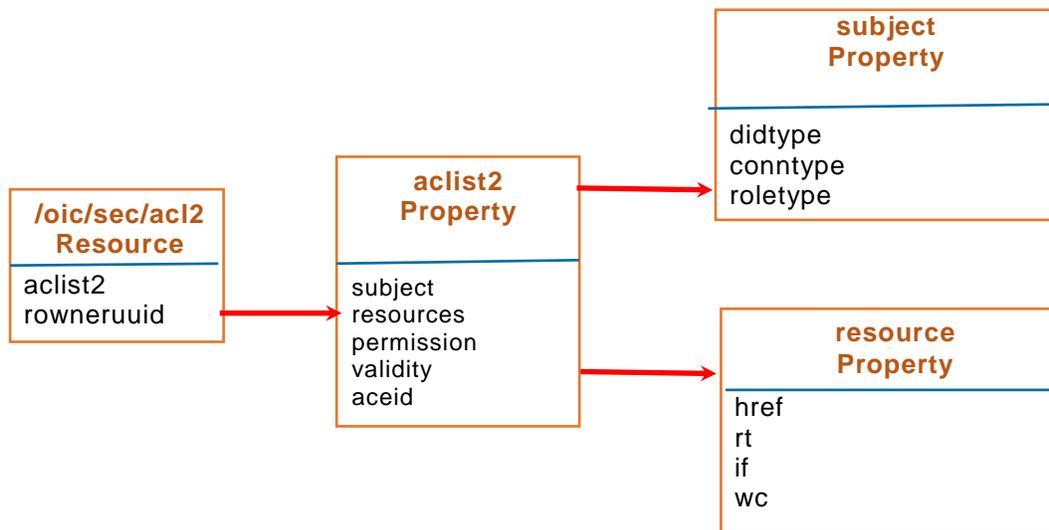
Figure 34 – OCF Security Resources



3199

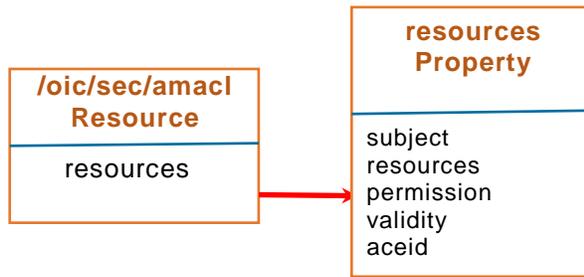
Figure 35 – /oic/sec/cred Resource and Properties

3200



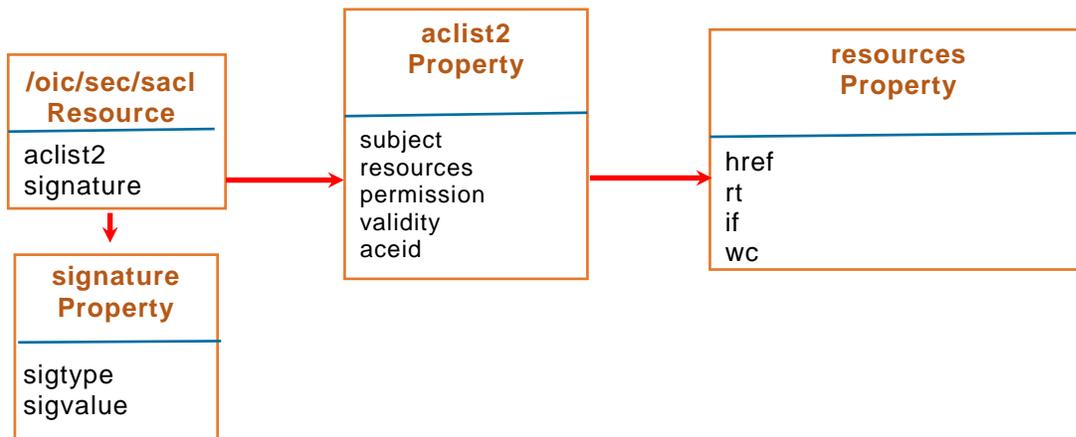
3201

Figure 36 – /oic/sec/acl2 Resource and Properties



3202

Figure 37 – /oic/sec/amacl Resource and Properties



3203

Figure 38 – /oic/sec/sacl Resource and Properties

3204 **13.2 Device Owner Transfer Resource**

3205 **13.2.1 Device Owner Transfer Resource General**

3206 The “/oic/sec/doxm” Resource contains the set of supported Device OTMs.

3207 Resource discovery processing respects the CRUDN constraints supplied as part of the security
 3208 Resource definitions contained in this document.

3209 “/oic/sec/doxm” Resource is defined in Table 24.

3210

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

3211 Table 25 defines the Properties of the "/oic/sec/doxm" Resource.

3212

Table 25 – Properties of the /oic/sec/doxm Resource

Property Title	Property Name	Value Type	Value Rule	Mandatory	Device State	Access Mode	Description
OTM	oxms	oic.sec.doxmtype	array	Yes		R	Value identifying the owner-transfer-method and the organization that defined the method.
OTM Selection	oxmsel	oic.sec.doxmtype	UINT16	Yes	RESET	R	Server shall set to (4) "oic.sec.oxm.self"
					RFOTM	RW	DOTS shall set to it's selected DOTS and both parties execute the DOTS. After secure owner transfer session is established DOTS shall update the oxmsel again making it permanent. If the DOTSDOXS fails the Server shall transition device state to RESET.
					RFPRO	R	n/a
					RFNOP	R	n/a
					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.

3213 **Table 26** defines the Properties of the "/oic/sec/didtype".

3214 **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

3215 The oxms Property contains a list of OTM where the entries appear in the order of preference.
3216 This Property contains the higher priority methods appearing before the lower priority methods.
3217 The DOTS queries this list at the time of onboarding and selects the most appropriate method.

3218 The DOTS shall update the oxmsel Property of the /oic/sec/doxm Resource with the OTM that
3219 was used to onboard the Device.

3220 OTMs consist of two parts, a URI identifying the vendor or organization and the specific method.

```

3221 <DoxmType> ::= <NSS>
3222 <NSS> ::= <Identifier> | { {<NID>"."} <NameSpaceQualifier> "." } <Method>
3223 <NID> ::= <Vendor-or-Organization>
3224 <Identifier> ::= INTEGER
3225 <NameSpaceQualifier> ::= String
3226 <Method> ::= String
3227 <Vendor-Organization> ::= String

```

3228 When an OTM successfully completes, the *owned* Property is set to '1' (TRUE). Consequently,
3229 subsequent attempts to take ownership of the Device will fail.

3230 The Server shall expose a persistent or semi-persistent a deviceuuid Property that is stored in
3231 the "/oic/sec/doxm" Resource when the devowneruuid Property of the "/oic/sec/doxm" Resource
3232 is UPDATED to non-nil UUID value.

3233 The DOTS should RETRIEVE the updated deviceuuid Property of the "/oic/sec/doxm" Resource
3234 after it has updated the devowneruuid Property value of the /oic/sec/doxm Resource to a non-nil-
3235 UUID value.

3236 The Device vendor shall determine that the Device identifier (deviceuuid) is persistent (not
3237 updatable) or that it is non-persistent (updatable by the owner transfer service – a.k.a DOTS).

3238 If the deviceuuid Property of "/oic/sec/doxm" Resource is persistent, the request to UPDATE shall
3239 fail with the error PROPERTY_NOT_FOUND.

3240 If the deviceuuid Property of the "/oic/sec/doxm" Resource is non-persistent, the request to
3241 UPDATE shall succeed and the value supplied by DOTS shall be remembered until the device is
3242 RESET. If the UPDATE to deviceuuid Property of the /oic/sec/doxm Resource fails while in the
3243 RFOTM Device state the device state shall transition to RESET where the Server shall set the
3244 value of the deviceuuid Property of the "/oic/sec/doxm" Resource to the nil-UUID (e.g.
3245 "00000000-0000-0000-0000-000000000000").

3246 Regardless of whether the device has a persistent or semi-persistent deviceuuid Property of the
3247 "/oic/sec/doxm" Resource, a temporary random UUID is exposed by the Server via the deviceuuid
3248 Property of the "/oic/sec/doxm" Resource each time the device enters RESET Device state. The
3249 temporary deviceuuid value is used while the device state is in the RESET state and while in the
3250 RFOTM device state until the DOTS establishes a secure OTM connection. The DOTS should
3251 RETRIEVE the updated deviceuuid Property value of the "/oic/sec/doxm" Resource after it has
3252 updated devowneruuid Property value of the "/oic/sec/doxm" Resource to a non-nil-UUID value.

3253 The deviceuuid Property of the "/oic/sec/doxm" Resource shall expose a persistent value(i.e. is
3254 not updatable via an OCF Interface) or a semi-persistent value (i.e. is updatable by the
3255 DOTSDOXs via an OCF Interface to the deviceuuid Property of the /oic/sec/doxm Resource
3256 during RFOTM Device state.).

3257 This temporary non-repeated value shall be exposed by the Device until the DOTS establishes a
3258 secure OTM connection and UPDATES the devowneruuid Property to a non-nil UUID value.
3259 Subsequently, (while in RFPRO, RFNOP and SRESET Device states) the deviceuuid Property of
3260 the /oic/sec/doxm Resource shall reveal the persistent or semi-persistent value to authenticated
3261 requestors and shall reveal the temporary non-repeated value to unauthenticated requestors.

3262 See 13.16 for additional details related to privacy sensitive considerations.

3263 **13.2.2 Persistent and Semi-persistent Device Identifiers**

3264 The Device vendor determines whether a device identifier can be set by a configuration tool or
3265 whether it is immutable. If it is an immutable value this document refers to it as a persistent
3266 device identifier. Otherwise, it is referred to as a semi-persistent device identifier. There are four
3267 device identifiers that could be considered persistent or semi-persistent :

3268 1) "deviceuuid" Property of "/oic/sec/doxm"

3269 2) "di" Property of "/oic/d"

3270 3) "piid" Property of "/oic/d"

3271 4) "pi" Property of "/oic/p"

3272 **13.2.3 Onboarding Considerations for Device Identifier**

3273 The deviceuuid is used to onboard the Device. The other identifiers (di, piid and pi) are not
3274 essential for onboarding. The onboarding service (aka DOTS) may not know a priori whether the
3275 Device to be onboarded is using persistent or semi-persistent identifiers. An OCF Security
3276 Domain owner may have a preference for persistent or semi-persistent device identifiers.
3277 Detecting whether the Device is using persistent or semi-persistent deviceuuid can be achieved
3278 by attempting to update it.

3279 If the "deviceuuid" Property of the /oic/sec/doxm Resource is persistent, then an UPDATE
 3280 request, at the appropriate time during onboarding shall fail with an appropriate error response.

3281 The appropriate time to attempt to update deviceuuid during onboarding exists when the Device
 3282 state is RFOTM and when devowneruuid Property value of the /oic/sec/doxm Resource has a
 3283 non-nil UUID value.

3284 If the "deviceuuid" Property of the /oic/sec/doxm Resource is semi-persistent, subsequent to a
 3285 successful UPDATE request to change it; the Device shall remember the semi-persistent value
 3286 until the next successful UPDATE request or until the Device state transitions to RESET.

3287 See 13.16 for addition behavior regarding "deviceuuid".

3288

3289 **13.2.4 OCF defined OTMs**

3290 Table 27 defines the Properties of the "oic.sec.doxmtype".

3291

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 ^a .
OCFSharedPin	oic.sec.doxm.rdp	1	The new Device randomly generates a PIN that is communicated via an out-of-band channel to a DOTSDOXS. An in-band Diffie-Hellman key agreement protocol establishes that both endpoints possess the PIN. Possession of the PIN by the DOTSDOXS signals the new Device that device ownership can be asserted.
OCFMfgCert	oic.sec.doxm.mfgcert	2	The new Device is presumed to have been manufactured with an embedded asymmetric private key that is used to sign a Diffie-Hellman exchange at Device onboarding. The manufacturer certificate should contain Platform hardening information and other security assurances assertions.
OCF Reserved	<Reserved>	3	Reserved
OCFSelf	oic.sec.doxm.self	4	The manufacturer shall set the /doxm.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

^a 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.

3292 **13.3 Credential Resource**

3293 **13.3.1 Credential Resource General**

3294 The /oic/sec/cred Resource maintains credentials used to authenticate the Server to Clients and
 3295 support services as well as credentials used to verify Clients and support services.

3296 Multiple credential types are anticipated by the OCF framework, including pair-wise pre-shared
 3297 keys, asymmetric keys, certificates and others. The credential Resource uses a Subject UUID to
 3298 distinguish the Clients and support services it recognizes by verifying an authentication challenge.

3299 In order to provide an interface which allows management of the "creds" Array Property, the
 3300 RETRIEVE, UPDATE and DELETE operations on the oic.r.cred Resource shall behave as follows:

3301 1) A RETRIEVE shall return the full Resource representation, except that any write-only
 3302 Properties shall be omitted (e.g. private key data).

3303 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
 3304 UPDATE request, as follows:

3305 a) If an UPDATE representation includes the "creds" array Property, then:

3306 i) Supplied creds with a "credid" that matches an existing "credid" shall replace
 3307 completely the corresponding cred in the existing "creds" array.

3308 ii) Supplied creds without a "credid" shall be appended to the existing "creds" array, and
 3309 a unique (to the cred Resource) "credid" shall be created and assigned to the new
 3310 cred by the Server. The "credid" of a deleted cred should not be reused, to improve
 3311 the determinism of the interface and reduce opportunity for race conditions.

3312 iii) Supplied creds with a "credid" that does not match an existing "credid" shall be
 3313 appended to the existing "creds" array, using the supplied "credid".

3314 iv) The rows in Table 29 corresponding to the "creds" array Property dictate the Device
 3315 States in which an UPDATE of the "creds" array Property is always rejected. If OCF
 3316 Device is in a Device State where the Access Mode in this row contains "R", then the
 3317 OCF Device shall reject all UPDATES of the "creds" array Property.

3318 3) A DELETE without query parameters shall remove the entire "creds" array, but shall not
 3319 remove the oic.r.cred Resource.

3320 4) A DELETE with one or more "credid" query parameters shall remove the cred(s) with the
 3321 corresponding credid(s) from the "creds" array.

3322 5) The rows in Table 29 corresponding to the "creds" array Property dictate the Device States in
 3323 which a DELETE is always rejected. If OCF Device is in a Device State where the Access
 3324 Mode in this row contains "R", then the OCF Device shall reject all DELETES.

3325 NOTE The oic.r.cred Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined in
 3326 ISO/IEC 30118-1:2018.

3327 "oic.r.cred" Resource is defined in Table 28.

3328 **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

3329 **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.

3331 All secure Device accesses shall have a /oic/sec/cred Resource that protects the end-to-end
3332 interaction.

3333 The /oic/sec/cred Resource shall be updateable by the service named in it's rowneruuid Property.

3334 ACLs naming /oic/sec/cred Resource should further restrict access beyond CRUDN access
3335 modes.

3336 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.crmstype	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.

3338 Table 31 defines the Properties of "oic.sec.credusagetype".

3339 **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

3340 Table 32 defines the Properties of "oic.sec.pubdatatype".

3341 **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

3342 Table 33 defines the Properties of "oic.sec.privdatatype".

3343 **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

3344 Table 34 defines the Properties of "oic.sec.optdatatype".

3345 **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

3346 Table 35 defines the Properties of "oic.sec.roletype".

3347 **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.

3348 **13.3.2 Properties of the Credential Resource**

3349 **13.3.2.1 Credential ID**

3350 Credential ID (credid) is a local reference to an entry in a creds Property array of the
3351 /oic/sec/cred Resource. The SRM generates it. The credid Property shall be used to
3352 disambiguate array elements of the creds Property.

3353 **13.3.2.2 Subject UUID**

3354 The subjectuuid Property identifies the Device to which an entry in a creds Property array of the
3355 /oic/sec/cred Resource shall be used to establish a secure session, verify an authentication
3356 challenge-response or to authenticate an authentication challenge.

3357 A subjectuuid Property that matches the Server's own deviceuuid Property, distinguishes the
3358 array entries in the creds Property that pertain to this Device.

3359 The subjectuuid Property shall be used to identify a group to which a group key is used to protect
3360 shared data.

3361 When certificate chain is used during secure connection establishment, the "subjectuuid"
3362 Property shall also be used to verify the identity of the responder. The presented certificate chain
3363 shall be accepted, if there is a matching Credential entry on the Device that satisfies all of the
3364 following:

- 3365 – Public Data of the entry contains trust anchor (root) of the presented chain.
- 3366 – Subject UUID of the entry matches UUID in the Common Name field of the End-Entity
3367 certificate in the presented chain. If Subject UUID of the entry is set as a wildcard "*", this
3368 condition is automatically satisfied.
- 3369 – Credential Usage of the entry is "oic.sec.cred.trustca".

3370 **13.3.2.3 Role ID**

3371 The roleid Property identifies a role that has been granted to the credential.

3372 **13.3.2.4 Credential Type**

3373 The credtype Property is used to interpret several of the other Property values whose contents
3374 can differ depending on credential type. These Properties include publicdata, privatedata and
3375 optionaldata. The credtype Property value of '0' ("no security mode") is reserved for testing and
3376 debugging circumstances. Production deployments shall not allow provisioning of credentials of
3377 type '0'. The SRM should introduce checking code that prevents its use in production
3378 deployments.

3379 **13.3.2.5 Public Data**

3380 The publicdata Property contains information that provides additional context surrounding the
3381 issuance of the credential. For example, it might contain information included in a certificate or
3382 response data from a CMS. It might contain wrapped data.

3383 **13.3.2.6 Private Data**

3384 The privatedata Property contains secret information that is used to authenticate a Device,
3385 protect data or verify an authentication challenge-response.

3386 The privatedata Property shall not be disclosed outside of the SRM's trusted computing perimeter.
3387 A secure element (SE) or trusted execution environment (TEE) should be used to implement the
3388 SRM's trusted computing perimeter. The privatedata contents may be referenced using a handle;
3389 for example if used with a secure storage sub-system.

3390 **13.3.2.7 Optional Data**

3391 The optionaldata Property contains information that is optionally supplied, but facilitates key
3392 management, scalability or performance optimization.

3393 **13.3.2.8 Period**

3394 The period Property identifies the validity period for the credential. If no validity period is
3395 specified the credential lifetime is undetermined. Constrained devices that do not implement a
3396 date-time capability shall obtain current date-time information from its CMS.

3397 **13.3.2.9 Credential Refresh Method Type Definition**

3398 The CMS shall implement the credential refresh methods specified in the crms Property of the
3399 oic.sec.creds array in the /oic/sec/cred Resource.

3400 Table 36 defines the values of "oic.sec.crmtime".

Table 36 – Value Definition of the oic.sec.crctype 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.rowneruuid 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.rowneruuid 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.rowneruuid 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.rowneruuid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruuid 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.rowneruuid Resource to identify the key management service that supports this credential refresh method. The Server uses its /oic/sec/cred.rowneruuid Resource to identify a key management service that supports this credential refresh method.

3402 13.3.2.10 Credential Usage

3403 Credential Usage indicates to the Device the circumstances in which a credential should be used.
3404 Five values are defined:

- 3405 – oic.sec.cred.trustca: This certificate is a trust anchor for the purposes of certificate chain
3406 validation, as defined in 10.3.
- 3407 – oic.sec.cred.cert: This credusage is used for certificates for which the Device possesses the
3408 private key and uses it for identity authentication in a secure session, as defined in clause
3409 10.4.
- 3410 – oic.sec.cred.rolecert: This credusage is used for certificates for which the Device possesses
3411 the private key and uses to assert one or more roles, as defined in clause 10.4.2.
- 3412 – oic.sec.cred.mfgtrustca: This certificate is a trust anchor for the purposes of the Manufacturer
3413 Certificate Based OTM as defined in clause 7.3.6.
- 3414 – oic.sec.cred.mfgcert: This certificate is used for certificates for which the Device possesses
3415 the private key and uses it for authentication in the Manufacturer Certificate Based OTM as
3416 defined in clause 7.3.6.

3417 **13.3.3 Key Formatting**

3418 **13.3.3.1 Symmetric Key Formatting**

3419 Symmetric keys shall have the format described in Table 37 and Table 38.

3420 **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.

3421

3422 **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.

3423 **13.3.3.2 Asymmetric Keys**

3424 Asymmetric key formatting is not available in this revision of the document.

3425 **13.3.3.3 Asymmetric Keys with Certificate**

3426 Key formatting is defined by certificate definition.

3427 **13.3.3.4 Passwords**

3428 Password formatting is not available in this revision of the document.

3429 **13.3.4 Credential Refresh Method Details**

3430 **13.3.4.1 Provisioning Service**

3431 The resource owner identifies the provisioning service. If the Server determines a credential
3432 requires refresh and the other methods do not apply or fail, the Server will request re-
3433 provisioning of the credential before expiration. If the credential is allowed to expire, the Server
3434 should delete the Resource.

3435 **13.3.4.2 Pre-Shared Key**

3436 **13.3.4.2.1 Pre-Shared Key General**

3437 Using this mode, the current PSK is used to establish a Diffie-Hellmen session key in DTLS. The
3438 TLS_PRF is used as the key derivation function (KDF) that produces the new (refreshed) PSK.

3439 $PSK = TLS_PRF(MasterSecret, Message, length);$

3440 – MasterSecret – is the MasterSecret value resulting from the DTLS handshake using one of
3441 the above ciphersuites.

3442 – Message is the concatenation of the following values:

3443 – RM - Refresh method – I.e. "oic.sec.crm.psk"

3444 – Device ID_A is the string representation of the Device ID that supplied the DTLS
3445 ClientHello.

3446 – Device ID_B is the Device responding to the DTLS ClientHello message

3447 – Length of Message in bytes.

3448 Both Server and Client use the PSK to update the /oic/sec/cred Resource's privatedata Property.
3449 If Server initiated the credential refresh, it selects the new validity period. The Server sends the
3450 chosen validity period to the Client over the newly established DTLS session so it can update the
3451 corresponding credential Resource for the Server.

3452 **13.3.4.2.2 Random PIN**

3453 Using this mode, the current unexpired PIN is used to generate a PSK following IETF RFC 2898.
3454 The PSK is used during the Diffie-Hellman exchange to produce a new session key. The session
3455 key should be used to switch from PIN to PSK mode.

3456 The PIN is randomly generated by the Server and communicated to the Client through an out-of-
3457 band method. The OOB method used is out-of-scope.

3458 The pseudo-random function (PBKDF2) defined by IETF RFC 2898. PIN is a shared value used
3459 to generate a pre-shared key. The PIN-authenticated pre-shared key (PPSK) is supplied to a
3460 DTLS ciphersuite that accepts a PSK.

3461 $PPSK = PBKDF2(PRF, PIN, RM, Device\ ID, c, dkLen)$

3462 The PBKDF2 function has the following parameters:

- 3463 – PRF – Uses the DTLS PRF.
- 3464 – PIN – Shared between Devices.
- 3465 – RM - Refresh method – I.e. "oic.sec.crm.rdp"
- 3466 – Device ID – UUID of the new Device.
- 3467 – c – Iteration count initialized to 1000, incremented upon each use.
- 3468 – dkLen – Desired length of the derived PSK in octets.

3469 Both Server and Client use the PPSK to update the /oic/sec/cred Resource's PrivateData
3470 Property. If Server initiated the credential refresh, it selects the new validity period. The Server
3471 sends the chosen validity period to the Client over the newly established DTLS session so it can
3472 update its corresponding credential Resource for the Server.

3473 **13.3.4.2.3 SKDC**

3474 A DTLS session is opened to the Server where the /oic/sec/cred Resource has an rowneruid
3475 Property value that matches the a CMS that implements SKDC functionality and where the Client
3476 credential entry supports the oic.sec.crm.skdc credential refresh method. A ticket request
3477 message is delivered to the CMS and in response returns the ticket request. The Server updates
3478 or instantiates an /oic/sec/cred Resource guided by the ticket response contents.

3479 **13.3.4.2.4 PKCS10**

3480 A DTLS session is opened to the Server where the /oic/sec/cred Resource has an rowneruid
3481 Property value that matches the a CMS that supports the oic.sec.crm.pk10 credential refresh
3482 method. A PKCS10 formatted message is delivered to the service. After the refreshed certificate
3483 is issued, the CMS pushes the certificate to the Server. The Server updates or instantiates an
3484 /oic/sec/cred Resource guided by the certificate contents.

3485 **13.3.4.3 Resource Owner**

3486 The Resource Owner Property allows credential provisioning to occur soon after Device
3487 onboarding before access to support services has been established. It identifies the entity
3488 authorized to manage the /oic/sec/cred Resource in response to Device recovery situations.

3489 **13.4 Certificate Revocation List**

3490 **13.4.1 CRL Resource Definition**

3491 Device certificates and private keys are kept in `cred` Resource. CRL is maintained and updated
 3492 with a separate `crl` Resource that is newly defined for maintaining the revocation list.

3493 "oic.r.crl" Resource is defined in Table 39.

3494 **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

3495 Table 40 defines the Properties of "oic.r.crl".

3496 **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

3497 **13.5 ACL Resources**

3498 **13.5.1 ACL Resources General**

3499 All Resource hosted by a Server are required to match an ACL policy. ACL policies can be
 3500 expressed using three ACL Resource Types: /oic/sec/acl2, /oic/sec/amacl and /oic/sec/sacl. The
 3501 subject (e.g. deviceuuid of the Client) requesting access to a Resource shall be authenticated
 3502 prior to applying the ACL check. Resources that are available to multiple Clients can be matched
 3503 using a wildcard subject. All Resources accessible via the unsecured communication endpoint
 3504 shall be matched using a wildcard subject.

3505 **13.5.2 OCF Access Control List (ACL) BNF defines ACL structures.**

3506 ACL structure in Backus-Naur Form (BNF) notation is defined in Table 41:

3507 **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>

3508 The <DeviceId> token means the requestor must possess a credential that uses <UUID> as its
3509 identity in order to match the requestor to the <ACE> policy.

3510 The <RoleId> token means the requestor must possess a role credential with <Character> as its
3511 role in order to match the requestor to the <ACE> policy.

3512 The <Wildcard> token "*" means any requestor is matched to the <ACE> policy, with or without
3513 authentication.

3514 When a <SubjectId> is matched to an <ACE> policy the <ResourceRef> is used to match the
3515 <ACE> policy to Resources.

3516 The <OIC_LINK> token contains values used to query existence of hosted Resources.

3517 The <Permission> token specifies the privilege granted by the <ACE> policy given the <SubjectId>
3518 and <ResourceRef> matching does not produce the empty set match.

3519 Permissions are defined in terms of CREATE ('C'), RETRIEVE ('R'), UPDATE ('U'), DELETE ('D'),
3520 NOTIFY ('N') and NIL ('-'). NIL is substituted for a permissions character that signifies the
3521 respective permission is not granted.

3522 The empty set match result defaults to a condition where no access rights are granted.

3523 If the <Validity> token exists, the <Permission> granted is constrained to the time <Period>.
3524 <Validity> may further be segmented into a <Recurrence> pattern where access may alternatively
3525 be granted and rescinded according to the pattern.

3526 13.5.3 ACL Resource

3527 There are two types of ACLs, 'acl' is a list of type 'ace' and 'acl2' is a list of type 'ace2'. A Device
3528 shall not host the /acl Resource.

3529 NOTE The /acl Resource is defined for backward compatibility and use by Provisioning Tools, etc.

3530 In order to provide an interface which allows management of array elements of the "aclist2"
3531 Property associated with an /oic/sec/acl2 Resource. The RETRIEVE, UPDATE and DELETE
3532 operations on the /oic/sec/acl2 Resource SHALL behave as follows:

- 3533 1) A RETRIEVE shall return the full Resource representation.
- 3534 2) An UPDATE shall replace or add to the Properties included in the representation sent with the
3535 UPDATE request, as follows:
 - 3536 a) If an UPDATE representation includes the array Property, then:
 - 3537 i) Supplied ACEs with an "aceid" that matches an existing "aceid" shall replace
3538 completely the corresponding ACE in the existing "aces2" array.
 - 3539 ii) Supplied ACEs without an "aceid" shall be appended to the existing "aces2" array,
3540 and a unique (to the acl2 Resource) "aceid" shall be created and assigned to the new
3541 ACE by the Server. The "aceid" of a deleted ACE should not be reused, to improve
3542 the determinism of the interface and reduce opportunity for race conditions.

3543 iii) Supplied ACEs with an "aceid" that does not match an existing "aceid" shall be
3544 appended to the existing "aces2" array, using the supplied "aceid".

3545 The rows in Table 47 defines the Properties of "oic.sec.acl2".

3546 iv) Table 47 corresponding to the "aclist2" array Property dictate the Device States in
3547 which an UPDATE of the "aclist2" array Property is always rejected. If OCF Device is
3548 in a Device State where the Access Mode in this row contains "R", then the OCF
3549 Device shall reject all UPDATES of the "aclist2" array Property.

3550 3) A DELETE without query parameters shall remove the entire "aces2" array, but shall not
3551 remove the oic.r.ace2 Resource.

3552 4) A DELETE with one or more "aceid" query parameters shall remove the ACE(s) with the
3553 corresponding aceid(s) from the "aces2" array.

3554 The rows in Table 47 defines the Properties of "oic.sec.acl2".

3555 5) Table 47 corresponding to the "aclist2" array Property dictate the Device States in which a
3556 DELETE is always rejected. If OCF Device is in a Device State where the Access Mode in this
3557 row contains "R", then the OCF Device shall reject all DELETES.

3558 NOTE The oic.r.acl2 Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined
3559 in ISO/IEC 30118-1:2018.

3560 Evaluation of local ACL Resource completes when all ACL Resource have been queried and no
3561 entry can be found for the requested Resource for the requestor – e.g. /oic/sec/acl, /oic/sec/sacl
3562 and /oic/sec/amacl do not match the subject and the requested Resource.

3563 It is possible the AMS has an ACL policy that satisfies a resource access request, but the
3564 necessary ACE has not been provisioned to Server. The Server may open a secure connection to
3565 the AMS to request ACL provisioning. The Server may use filter criteria that returns a subset of
3566 the AMS ACL policy. The AMS shall obtain the Server Device ID using the secure connection
3567 context.

3568 The AMS maintains an AMACL policy for Servers it manages. If the Server connects to the AMS
3569 to process an /oic/sec/amacl Resource. The AMS shall match the AMACL policy and return the
3570 Permission Property or an error if no match is found.

3571 If the requested Resource is still not matched, the Server returns an error. The requester should
3572 query the Server to discover the configured AMS services. The Client should contact the AMS to
3573 request a sacl (/oic/sec/sacl) Resource. Performing the following operations implement this type
3574 of request:

3575 1) Client: Open secure connection to AMS.

3576 2) Client: RETRIEVE /oic/sec/acl2?deviceuuid="XXX...",resources="href"

3577 3) AMS: constructs a /oic/sec/sacl Resource that is signed by the AMS and returns it in
3578 response to the RETRIEVE command.

3579 4) Client: UPDATE /oic/sec/sacl [{ ...sacl... }]

3580 5) Server: verifies sacl signature using AMS credentials and installs the ACL Resource if valid.

3581 6) Client: retries original Resource access request. This time the new ACL is included in the
3582 local ACL evaluation.

3583 The ACL contained in the /oic/sec/sacl Resource should grant longer term access that satisfies
3584 repeated Resource requests.

3585 "oic.r.acl" Resource is defined in Table 42.

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

3587 **Table 43** defines the Properties of "oic.r.acl ".**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 rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
					R	RFNOP	Access to NCRs is permitted after a matching ACE is found.
					RW	SRESET	The DOTS (referenced via devowneruuid Property of /oic/sec/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	rowneruuid	String	uuid	Yes	N/A	N/A	The resource owner Property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
					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 rowneruuid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a

					RW	SRESET	The DOTS (referenced via /doxm devowneruid Property or the /doxm rowneruid Property) 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.
--	--	--	--	--	----	--------	---

3589 Table 44 defines the Properties of "oic.r.ace".

3590 **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.crudntype	bitmask	RW	Yes	Bitmask encoding of CRUDN permission
Validity	validity	oic.sec.ace/definitions/time-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	subjectuid	String	uuid, "*"	RW	Yes	A uuid that identifies the Device to which this ACE applies to or "*" for anonymous access.

3591 Table 45 defines the values of "oic.sec.crudntype".

3592 **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	RETRIEVE, 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

3593 "oic.sec.acl2" Resource is defined in Table 28.

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

3595 Table 47 defines the Properties of "oic.sec.acl2".

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 rowneruuid property) shall update the aclist entries after mutually authenticated secure session is established. Access to NCRs is prohibited.
			RFNOP	R	Access to NCRs is permitted after a matching ACE2 is found.
			SRESET	RW	The DOTS (referenced via devowneruuid Property of /oic/sec/doxm Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
rowneruuid	uuid	Yes	N/A		The resource owner Property (rowneruuid) is used by the Server to reference a service provider trusted by the Server. Server shall verify the service provider is authorized to perform the requested action
			RESET	R	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
			RFOTM	RW	The DOTS should configure the rowneruuid Property of /oic/sec/acl2 Resource when a successful owner transfer session is established.
			RFPRO	R	n/a
			RFNOP	R	n/a
			SRESET	RW	The DOTS (referenced via devowneruuid Property or rowneruuid Property of /oic/sec/doxm Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruuid Property does not refer to a valid DOTS the Server shall transition to RESET device state.

3598 Table 48 defines the Properties of "oic.sec.ace2".

3599 **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.bitm ask	Yes	Bitmask encoding of CRUDN permission
validity	array of oic.sec.time- pattern	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the IETF RFC 5545 Period, and a string array representing a recurrence rule using the IETF RFC 5545 Recurrence.
aceid	integer	Yes	An aceid is unique with respect to the array entries in the aclist2 Property.

3600 Table 49 defines the Properties of "oic.sec.ace2.resource-ref".

3601 **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.

3602 Table 50 defines the values of "oic.sec.ace2.resource-ref".

3603 **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

3604 Local ACL Resources supply policy to a Resource access enforcement point within an OCF stack
3605 instance. The OCF framework gates Client access to Server Resources. It evaluates the subject's
3606 request using policies contained in ACL resources.

3607 Resources named in the ACL policy can be fully qualified or partially qualified. Fully qualified
3608 Resource references include the device identifier in the href Property that identifies the remote
3609 Resource Server that hosts the Resource. Partially qualified references means the local
3610 Resource Server hosts the Resource. If a fully qualified resource reference is given, the
3611 Intermediary enforcing access shall have a secure channel to the Resource Server and the

3612 Resource Server shall verify the Intermediary is authorized to act on its behalf as a Resource
3613 access enforcement point.

3614 Resource Servers should include references to Device and ACL Resources where access
3615 enforcement is to be applied. However, access enforcement logic shall not depend on these
3616 references for access control processing as access to Server Resources will have already been
3617 granted.

3618 Local ACL Resources identify a Resource Owner service that is authorized to instantiate and
3619 modify this Resource. This prevents non-terminating dependency on some other ACL Resource.
3620 Nevertheless, it should be desirable to grant access rights to ACL Resources using an ACL
3621 Resource.

3622 An ACE or ACE2 entry is called currently valid if the validity period of the ACE or ACE2 entry
3623 includes the time of the request. The validity period in the ACE or ACE2 may be a recurring time
3624 period (e.g., daily from 1:00-2:00). Matching the resource(s) specified in a request to the
3625 resource Property of the ACE or ACE2 is defined in Clause 12.2. For example, one way they can
3626 match is if the Resource URI in the request exactly matches one of the resource references in the
3627 ACE or ACE2 entries.

3628 A request will match an ACE if any of the following are true:

3629 1) The deviceuuid Property associated with the secure session matches the "subjectuid" of the
3630 ACE; AND the Resource of the request matches one of the resources Property of the ACE;
3631 AND the ACE is currently valid.

3632 2) The ACE subjectuid Property contains the wildcard "*" character; AND the Resource of the
3633 request matches one of the resources Property of the ACE; AND the ACE is currently valid.

3634 3) When authentication uses a symmetric key credential;

3635 AND the CoAP payload query string of the request specifies a role, which is associated with
3636 the symmetric key credential of the current secure session;

3637 AND the CoAP payload query string of the request specifies a role, which is contained in the
3638 oic.r.cred.creds.roleid Property of the current secure session;

3639 AND the resource of the request matches one of the resources Property of the ACE;

3640 AND the ACE is currently valid.

3641 A request will match an ACE2 if any of the following are true:

3642 1) The ACE2 subject Property is of type oic.sec.didtype has a UUID value that matches the
3643 deviceuuid Property associated with the secure session;

3644 AND the Resource of the request matches one of the resources Property of the ACE2
3645 oic.sec.ace2.resource-ref;

3646 AND the ACE2 is currently valid.

3647 2) The ACE2 subject Property is of type oic.sec.conntype and has the wildcard value that
3648 matches the currently established connection type;

3649 AND the resource of the request matches one of the resources Property of the ACE2
3650 oic.sec.ace2.resource-ref;

3651 AND the ACE2 is currently valid.

3652 3) When Client authentication uses a certificate credential;

3653 AND one of the roleid values contained in the role certificate matches the roleid Property of
3654 the ACE2 oic.sec.roletype;

3655 AND the role certificate public key matches the public key of the certificate used to establish
3656 the current secure session;

3657 AND the resource of the request matches one of the array elements of the resources Property
3658 of the ACE2 oic.sec.ace2.resource-ref;

3659 AND the ACE2 is currently valid.

3660 4) When Client authentication uses a certificate credential;

3661 AND the CoAP payload query string of the request specifies a role, which is member of the
3662 set of roles contained in the role certificate;

3663 AND the roleid values contained in the role certificate matches the roleid Property of the
3664 ACE2 oic.sec.roletype;

3665 AND the role certificate public key matches the public key of the certificate used to establish
3666 the current secure session;

3667 AND the resource of the request matches one of the resources Property of the ACE2
3668 oic.sec.ace2.resource-ref;

3669 AND the ACE2 is currently valid.

3670 5) When Client authentication uses a symmetric key credential;

3671 AND one of the roleid values associated with the symmetric key credential used in the secure
3672 session, matches the roleid Property of the ACE2 oic.sec.roletype;

3673 AND the resource of the request matches one of the array elements of the resources Property
3674 of the ACE2 oic.sec.ace2.resource-ref;

3675 AND the ACE2 is currently valid.

3676 6) When Client authentication uses a symmetric key credential;

3677 AND the CoAP payload query string of the request specifies a role, which is contained in the
3678 oic.r.cred.creds.roleid Property of the current secure session;

3679 AND CoAP payload query string of the request specifies a role that matches the roleid
3680 Property of the ACE2 oic.sec.roletype;

3681 AND the resource of the request matches one of the array elements of the resources Property
3682 of the ACE2 oic.sec.ace2.resource-ref;

3683 AND the ACE2 is currently valid.

3684 A request is granted if ANY of the 'matching' ACEs contains the permission to allow the
3685 request. Otherwise, the request is denied.

3686 There is no way for an ACE to explicitly deny permission to a resource. Therefore, if one Device
3687 with a given role should have slightly different permissions than another Device with the same
3688 role, they must be provisioned with different roles.

3689 The Server is required to verify that any hosted Resource has authorized access by the Client
3690 requesting access. The /oic/sec/acl2 Resource is co-located on the Resource host so that the
3691 Resource request processing should be applied securely and efficiently. See Annex A for
3692 example.

3693 **13.6 Access Manager ACL Resource**

3694 "oic.r.amacl" Resource is defined in Table 51.

3695

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

3696 Table 52 defines the Properties of "oic.r.amacl".

3697

Table 52 – Properties of the oic.r.amacl Resource

Property Title	Property Name	Value Type	Value Rule	Access Mode	Mandator y	Description
Resources	resources	oic.sec.ace2.resource-ref	array	RW	Yes	Multiple links to this host's Resources

3698 The AMS should be used to centralize management of access policy, but requires Servers to
 3699 open a connection to the AMS whenever the named Resources are accessed. See A.2 for
 3700 example.

3701 **13.7 Signed ACL Resource**

3702 "oic.r.sacl" Resource is defined in Table 53.

3703

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

3704 Table 54 defines the Properties of "oic.r.sacl".

3705

Table 54 – Properties of the oic.r.sacl Resource

Property Title	Property Name	Value Type	Value Rule	Mandator y	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 devowneruuid Property of /oic/sec/doxm Resource) should evaluate the integrity of and may update aclist entries when a secure session is established and the Server and DOTS are authenticated.
Signature	signature	oic.sec.sigtype	N/A	Yes	N/A	N/A	The signature over the ACL Resource

3706 Table 55 defines the Properties of "oic.sec.sigtype".

3707 **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

3708 **13.8 Provisioning Status Resource**

3709 The "/oic/sec/pstat" Resource maintains the Device provisioning status. Device provisioning
3710 should be Client-directed or Server-directed. Client-directed provisioning relies on a Client device
3711 to determine what, how and when Server Resources should be instantiated and updated. Server-
3712 directed provisioning relies on the Server to seek provisioning when conditions dictate. Server-
3713 directed provisioning depends on configuration of the rowneruuid Property of the /oic/sec/doxm,
3714 /oic/sec/cred and /oic/sec/acl2 Resources to identify the device ID of the trusted DOTS, CMS and
3715 AMS services respectively. Furthermore, the /oic/sec/cred Resource should be provisioned at
3716 ownership transfer with credentials necessary to open a secure connection with appropriate
3717 support service.

3718 "oic.r.pstat" Resource is defined in Table 56.

3719 **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

3720 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			Deprecated
Target Mode	tm	oic.sec.dpmttype	bitmask	Yes			Deprecated
Operational Mode	om	oic.sec.pomttype	bitmask	Yes	R	RESET	Server shall set to manufacturer default.
					RW	RFOTM	Set by DOTS after successful OTM
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by DOTS.
Supported Mode	sm	oic.sec.pomttype	bitmask	Yes	R	All states	Supported provisioning services operation modes
Device UUID	deviceuuid	String	uuid	Yes	RW	All states	[DEPRECATED] A uuid that identifies the Device to which the status applies
Resource Owner ID	rowneruid	String	uuid	Yes	R	RESET	Server shall set to the nil uuid value (e.g. "00000000-0000-0000-0000-000000000000")
					RW	RFOTM	The DOTS should configure the rowneruid Property when a successful owner transfer session is established.
					R	RFPRO	n/a
					R	RFNOP	n/a
					RW	SRESET	The DOTS (referenced via devowneruid Property of /oic/sec/doxm Resource) should verify and if needed, update the resource owner Property when a mutually authenticated secure session is established. If the rowneruid does not refer to a valid DOTS the Server shall transition to RESET Device state.

3722 The provisioning status Resource /oic/sec/pstat is used to enable Devices to perform self-
3723 directed provisioning. Devices are aware of their current configuration status and a target
3724 configuration objective. When there is a difference between current and target status, the Device

3725 should consult the rowneruid Property of /oic/sec/cred Resource to discover whether any
 3726 suitable provisioning services exist. The Device should request provisioning if configured to do so.
 3727 The om Property of /oic/sec/pstat Resource will specify expected Device behaviour under these
 3728 circumstances.

3729 Self-directed provisioning enables Devices to function with greater autonomy to minimize
 3730 dependence on a central provisioning authority that should be a single point of failure in the OCF
 3731 Security Domain.

3732 Table 58 defines the Properties of "/oic/sec/dostype".

3733 **Table 58 – Properties of the /oic/sec/dostype Property**

Property Title	Property Name	Value Type	Value Rule	Mandator y	Access Mode	Device State	Description
Device Onboarding State	s	UINT16	enum (0=RESET, 1=RFOTM, 2=RFPRO, 3=RFNOP, 4=SRESET	Y	R	RESET	The Device is in a hard reset state.
					RW	RFOTM	Set by DOTS after successful OTM to RFPRO.
					RW	RFPRO	Set by CMS, AMS, DOTS after successful authentication
					RW	RFNOP	Set by CMS, AMS, DOTS after successful authentication
					RW	SRESET	Set by CMS, AMS, DOTS after successful authentication
Pending state	p	Boolean	T F	Y	R	All States	TRUE (1) – ‘s’ state is pending until all necessary changes to Device resources are complete FALSE (0) – ‘s’ state changes are complete

3734 In all Device states:

3735 – An authenticated and authorised Client may change the Device state of a Device by updating
 3736 pstat.dos.s to the desired value. The allowed Device state transitions are defined in
 3737 Figure 27.

3738 – Prior to updating pstat.dos.s, the Client configures the Device to meet entry conditions for the
 3739 new Device state. The SVR definitions define the entity (Client or Server) expected to
 3740 perform the specific SVR configuration change to meet the entry conditions. Once the Client
 3741 has configured the aspects for which the Client is responsible, it may update pstat.dos.s. The
 3742 Server then makes any changes for which the Server is responsible, including updating
 3743 required SVR values, and set pstat.dos.s to the new value.

3744 – The pstat.dos.p Property is read-only by all Clients.

3745 – The Server sets pstat.dos.p to TRUE before beginning the process of updating pstat.dos.s,
 3746 and sets it back to FALSE when the pstat.dos.s change is completed.

3747 Any requests to update pstat.dos.s while pstat.dos.p is TRUE are denied.

3748 When Device state is RESET:

3749 – All SVR content is removed and reset to manufacturer default values.

3750 – The default manufacturer Device state is RESET.

3751 – NCRs are reset to manufacturer default values.

3752 – NCRs are inaccessible.

- 3753 – After successfully processing RESET the SRM transitions to RFOTM by setting s Property of
3754 /oic/sec/dostype Resource to RFOTM.
- 3755 When Device state is RFOTM:
- 3756 – NCRs are inaccessible.
- 3757 – Before OTM is successful, the deviceuuid Property of /oic/sec/doxm Resource shall be set to
3758 a temporary non-repeated value as defined in clauses 13.2 and 13.16.
- 3759 – Before OTM is successful, the s Property of /oic/sec/dostype Resource is read-only by
3760 unauthenticated requestors
- 3761 – After the OTM is successful, the s Property of /oic/sec/dostype Resource is read-write by
3762 authorized requestors.
- 3763 – The negotiated Device OC is used to create an authenticated session over which the DOTS
3764 directs the Device state to transition to RFPRO.
- 3765 – If an authenticated session cannot be established the ownership transfer session should be
3766 disconnected and SRM sets back the Device state to RESET state.
- 3767 – Ownership transfer session, especially Random PIN OTM, should not exceed 60 seconds, the
3768 SRM asserts the OTM failed, should be disconnected, and transitions to RESET
3769 (/pstat.dos.s=RESET).
- 3770 – The DOTS UPDATES the devowneruuid Property in the /doxm Resource to a non-nil UUID
3771 value. The DOTSDOXS (or other authorized client) may update it multiple times while in
3772 RFOTM. It is not updatable while in other device states except when the Device state returns
3773 to RFOTM through RESET.
- 3774 – The DOTS may have additional provisioning tasks to perform while in RFOTM. When done,
3775 the DOTSDOXS UPDATES the "owned" Property in the /doxm Resource to "true".
- 3776 When Device state is RFPRO:
- 3777 – The s Property of /oic/sec/dostype Resource is read-only by unauthorized requestors and
3778 read-write by authorized requestors.
- 3779 – NCRs are inaccessible, except for Easy Setup Resources, if supported.
- 3780 – The OCF Server may re-create NCRs.
- 3781 – An authorized Client may provision SVRs as needed for normal functioning in RFNOP.
- 3782 – An authorized Client may perform consistency checks on SVRs to determine which shall be
3783 re-provisioned.
- 3784 – Failure to successfully provision SVRs may trigger a state change to RESET. For example, if
3785 the Device has already transitioned from SRESET but consistency checks continue to fail.
- 3786 – The authorized Client sets the /pstat.dos.s=RFNOP.
- 3787 When Device state is RFNOP:
- 3788 – The /pstat.dos.s Property is read-only by unauthorized requestors and read-write by
3789 authorized requestors.
- 3790 – NCRs, SVRs and core Resources are accessible following normal access processing.
- 3791 – An authorized may transition to RFPRO. Only the Device owner may transition to SRESET or
3792 RESET.
- 3793 When Device state is SRESET:
- 3794 – NCRs are inaccessible. The integrity of NCRs may be suspect but the SRM doesn't attempt to
3795 access or reference them.

- 3796 – SVR integrity is not guaranteed, but access to some SVR Properties is necessary. These
- 3797 include devowneruuid Property of the "/oic/sec/doxm" Resource,
- 3798 "creds":[{"subjectuuid":<devowneruuid>},...] Property of the /oic/sec/cred Resource and s
- 3799 Property of the /oic/sec/dostype Resource of /oic/sec/pstat Resource.
- 3800 – The certificates that identify and authorize the Device owner are sufficient to re-create
- 3801 minimalist /cred and /doxm resources enabling Device owner control of SRESET. If the SRM
- 3802 can't establish these Resources, then it will transition to RESET state.
- 3803 – An authorized Client performs SVR consistency checks. The caller may provision SVRs as
- 3804 needed to ensure they are available for continued provisioning in RFPRO or for normal
- 3805 functioning in RFNOP.
- 3806 – The authorized Device owner may avoid entering RESET state and RFOTM by UPDATING
- 3807 dos.s Property of the /pstat Resource with RFPRO or RFNOP values
- 3808 – ACLs on SVR are presumed to be invalid. Access authorization is granted according to
- 3809 Device owner privileges.
- 3810 – The SRM asserts a Client-directed operational mode (e.g. /pstat.om=CLIENT_DIRECTED).
- 3811 The *provisioning mode* type is a 16-bit mask enumerating the various Device provisioning modes.
- 3812 "{ProvisioningMode}" should be used in this document to refer to an instance of a provisioning
- 3813 mode without selecting any particular value.
- 3814 "oic.sec.dpmttype" is defined in Table 59.

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

3816 Table 60 and Table 61 define the values of "oic.sec.dpmttype".

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)

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

3819 The *provisioning operation mode* type is a 8-bit mask enumerating the various provisioning

3820 operation modes.

3821 "oic.sec.pomtype" is defined in Table 62.

3822 **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

3823 Table 63 defines the values of "oic.sec.pomtype".

3824 **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

3825 **13.9 Certificate Signing Request Resource**

3826 The /oic/sec/csr Resource is used by a Device to provide its desired identity, public key to be
 3827 certified, and a proof of possession of the corresponding private key in the form of a IETF RFC
 3828 2986 PKCS#10 Certification Request. If the Device supports certificates (i.e. the sct Property of
 3829 /oic/sec/doxm Resource has a 1 in the 0x8 bit position), the Device shall have a /oic/sec/csr
 3830 Resource.

3831 "oic.r.csr" Resource is defined in Table 64.

3832 **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

3833 Table 65 defines the Properties of "oic.r.csr".

3834 **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
----------	----------	--------	---	-----	---

3835 The Device chooses which public key to use, and may optionally generate a new key pair for this
3836 purpose.

3837 In the CSR, the Common Name component of the Subject Name shall contain a string of the
3838 format "uuid:X" where X is the Device's requested UUID in the format defined by IETF RFC 4122.
3839 The Common Name, and other components of the Subject Name, may contain other data. If the
3840 Device chooses to include additional information in the Common Name component, it shall delimit
3841 it from the UUID field by white space, a comma, or a semicolon.

3842 If the Device does not have a pre-provisioned key pair to use, but is capable and willing to
3843 generate a new key pair, the Device may begin generation of a key pair as a result of a
3844 RETRIEVE of this resource. If the Device cannot immediately respond to the RETRIEVE request
3845 due to time required to generate a key pair, the Device shall return an "operation pending" error.
3846 This indicates to the Client that the Device is not yet ready to respond, but will be able at a later
3847 time. The Client should retry the request after a short delay.

3848 13.10 Roles Resource

3849 The roles Resource maintains roles that have been asserted with role certificates, as described in
3850 Clause 10.4.2. Asserted roles have an associated public key, i.e., the public key in the role
3851 certificate. Servers shall only grant access to the roles information associated with the public key
3852 of the Client. The roles Resource should be viewed as an extension of the (D)TLS session state.
3853 See 10.4.2 for how role certificates are validated.

3854 The roles Resource shall be created by the Server upon establishment of a secure (D)TLS
3855 session with a Client, if is not already created. The roles Resource shall only expose a secured
3856 OCF Endpoint in the /oic/res response. A Server shall retain the roles Resource at least as long
3857 as the (D)TLS session exists. A Server shall retain each certificate in the roles Resource at least
3858 until the certificate expires or the (D)TLS session ends, whichever is sooner. The requirements of
3859 clause 10.3 and 10.4.2 to validate a certificate's time validity at the point of use always apply. A
3860 Server should regularly inspect the contents of the roles resource and purge contents based on a
3861 policy it determines based on its resource constraints. For example, expired certificates, and
3862 certificates from Clients that have not been heard from for some arbitrary period of time could be
3863 candidates for purging.

3864 The roles Resource is implicitly created by the Server upon establishment of a (D)TLS session. In
3865 more detail, the RETRIEVE, UPDATE and DELETE operations on the roles Resource shall
3866 behave as follows. Unlisted operations are implementation specific and not reliable.

- 3867 1) A RETRIEVE request shall return all previously asserted roles associated with the currently
3868 connected and authenticated Client's identity. RETRIEVE requests with a "credid" query
3869 parameter is not supported; all previously asserted roles associated with the currently
3870 connected and authenticated Client's identity are returned.
- 3871 2) An UPDATE request that includes the "roles" Property shall replace or add to the Properties
3872 included in the array as follows:
 - 3873 a) If either the "publicdata" or the "optionaldata" are different than the existing entries in the
3874 "roles" array, the entry shall be added to the "roles" array with a new, unique "credid"
3875 value.

3876 b) If both the "publicdata" and the "optionaldata" match an existing entry in the "roles" array,
 3877 the entry shall be considered to be the same. The Server shall reply with a 2.04 Changed
 3878 response and a duplicate entry shall not be added to the array.

3879 c) The "credid" Property is optional in an UPDATE request and if included, it may be ignored
 3880 by the Server. The Server shall assign a unique "credid" value for every entry of the
 3881 "roles" array.

3882 3) A DELETE request without a "credid" query parameter shall remove all entries from the
 3883 "/oic/sec/roles" resource array corresponding to the currently connected and authenticated
 3884 Client's identity.

3885 4) A DELETE request with a "credid" query parameter shall remove only the entries of the
 3886 /oic/sec/roles resource array corresponding to the currently connected and authenticated
 3887 Client's identity and where the corresponding "credid" matches the entry.

3888 NOTE The oic.r.roles Resource's use of the DELETE operation is not in accordance with the OCF Interfaces defined
 3889 in ISO/IEC 30118-1:2018.

3890 "oic.r.roles" Resource is defined in Table 66.

3891 **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

3892 Table 67 defines the Properties of "oic.r.roles".

3893 **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

3894 Because oic.r.roles shares the oic.sec.cred schema with oic.r.cred, "subjectuud" is a required Property. However,
 3895 "subjectuud" is not used in a role certificate. Therefore, a Device may ignore the "subjectuud" Property if the Property
 3896 is contained in an UPDATE request to the /oic/sec/roles Resource.

3897 **13.11 Account Resource**

3898 The Account Resource specifies the Properties based on IETF RFC 6749 Access Token based
 3899 account creation. The mechanism to obtain credentials is described in Clause 7.5. The Account
 3900 Resource is used for Device Registration. The Account Resource is instantiated on the OCF
 3901 Cloud as "oic/sec/account" SVR and is used by cloud-enabled Devices to register with the OCF
 3902 Cloud. It should be only accessible on a secure channel; non-secure channel should not be able
 3903 access this Resource.

3904 During the Device Registration process, an OCF Cloud can provide a distinct URI of another OCF
 3905 Cloud ("redirected-to" OCF Cloud). Both initial and redirected-to OCF Clouds are expected to
 3906 belong to the same Vendor; they are assumed to have the same UUID and are assumed to have
 3907 an out-of-band communication mechanism established. Device does not have to perform the
 3908 Device Registration on the redirected-to OCF Cloud and the OCF Cloud may ignore such
 3909 attempts. Redirected-to OCF Cloud is expected to accept the Access Token, provided to the
 3910 Device by the initial OCF Cloud.

3911 The "di", "uid", "refreshtoken" and "accesstoken" Properties of the Account Resource should be
 3912 securely stored as described in Clause 15.

3913 The RETRIEVE operation on OCF Cloud's "/oic/sec/account" Resource is not allowed and the
 3914 OCF Cloud is expected to reject all attempts to perform such operation.

3915 The UPDATE operation on the OCF Cloud's "/oic/sec/account" Resource behaves as follows:

- 3916 – A Device intending to register with the OCF Cloud shall send UPDATE with following
 3917 Properties “di” (“di” Property Value of “/oic/d” Resource), and “accesstoken” as configured by
 3918 the Mediator (“at” Property Value of oic.r.coapcloudconf Resource). The OCF Cloud verifies it
 3919 is the same “accesstoken” which was assigned to the Mediator for the corresponding “di”
 3920 Property Value. The “accesstoken” is the permission for the Device to access the OCF Cloud.
 3921 If the “apn” was included when the Mediator UPDATED the “oic.r.coapcloudconf” Resource,
 3922 the Device shall also include “authprovider” Property when registering with the OCF Cloud. If
 3923 no “apn” is specified, then the “authprovider” Property shall not be included in the UPDATE
 3924 request.
- 3925 – OCF Cloud returns “accesstoken”, “uid”, “refresh token”, “expiresin” It may also return
 3926 “redirecturi”. Received “accesstoken” is to be treated by Device as an Access Token with
 3927 “Bearer” token type as defined in IETF RFC 6750. This “accesstoken” shall be used for the
 3928 following Account Session start using “/oic/sec/session” SVR. Received “refresh token” is to be
 3929 treated by Device as a Refresh Token as defined in IETF RFC 6749. The Device stores the
 3930 OCF Cloud's Response values. If “redirecturi” is received, Device shall use received value as
 3931 a new OCF Cloud URI instead of “cis” Property Value of “oic.r.coapcloudconf” Resource for
 3932 further connections.

3933 The DELETE operation on the OCF Cloud's "/oic/sec/account" Resource should behave as
 3934 follows:

- 3935 – To deregister with the OCF Cloud, a DELETE operation shall be sent with the “accesstoken”
 3936 and either “uid”, or “di” to be deregistered with the OCF Cloud. On DELETE with the OCF
 3937 Cloud, the Device should also delete values internally stored. Once deregister with an OCF
 3938 Cloud, Device can connect to any other OCF Cloud. Device deregistered need to go through
 3939 the steps in 7.5 again to be registered with the OCF Cloud.

3940 " oic.r.account " Resource is defined in Table 68.

3941 **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.baseline	Resource used for a device to add itself under a given credential	N/A

3942 Table 69 defines the Properties of "oic.r.account ".

3943 **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.

3944 **13.12 Account Session resource**

3945 The "/oic/sec/session" Resource hosted on the OCF Cloud is used for creating connections with
 3946 the OCF Cloud subsequent to Device registration through "/oic/sec/account" Resource. The
 3947 "/oic/sec/session" Resource requires the device ID, User ID and Access Token which are stored
 3948 securely on the Device.

3949 The /oic/sec/session Resource is exposed by the OCF Cloud. It should be only accessible on a
 3950 secure channel; non-secure channel cannot access this Resource.

3951 The RETRIEVE operation on OCF Cloud's "/oic/sec/session" Resource is not allowed and the
 3952 OCF Cloud is expected to reject all attempts to perform such operation.

3953 The UPDATE operation is defined as follows for OCF Cloud's "/oic/sec/session" Resource:

- 3954 – The Device connecting to the OCF Cloud shall send an UPDATE request message to the OCF
 3955 Cloud's /oic/sec/session Resource. The message shall include the "di" Property Value of
 3956 /oic/d Resource and "uid", "login" Value ("true" to establish connection; "false" to disconnect)
 3957 and "accesstoken" as returned by OCF Cloud during Device Registration. The OCF Cloud
 3958 verifies it is the same Access Token which was returned to the Device during Device
 3959 Registration process. If Device was attempting to establish the connection and provided
 3960 values were verified as correct by the OCF Cloud, OCF Cloud sends a response with
 3961 remaining lifetime of the associated Access Token ("expiresin" Property Value).

3962 "oic.r.session" Resource is defined in Table 70.

3963 **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.baseline	Resource that enables a device to manage its session using login or logout	N/A

3964 Table 71 defines the Properties of "oic.r.session".

3965 **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	acesstoken	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

3966 **13.13 Account Token Refresh Resource**

3967 The "/oic/sec/tokenrefresh" Resource is used by the Device for refreshing the Access Token.

3968 The "/oic/sec/tokenrefresh" Resource is hosted by the OCF Cloud. It should be only accessible
3969 on a secure channel; non-secure channel cannot access this Resource.

3970 The Device should use "/oic/sec/tokenrefresh" to refresh the Access Token with the OCF Cloud,
3971 when the time specified in "expiresin" is near.

3972 The RETRIEVE operation on OCF Cloud's "/oic/sec/ tokenrefresh" Resource is not allowed and
3973 the OCF Cloud is expected to reject all attempts to perform such operation.

3974 The UPDATE operation is defined as follows for "/oic/sec/tokenrefresh" Resource

- 3975 – The Device attempting to refresh the Access Token shall send an UPDATE request message
3976 to the OCF Cloud's /oic/sec/tokenrefresh Resource. The message shall include the "di"
3977 Property Value of /oic/d Resource, "uid" and "refreshtoken", as returned by OCF Cloud.
- 3978 – OCF Cloud response is expected to include a "refreshtoken", new "acesstoken", and
3979 "expiresin". Received "acesstoken" is to be treated by Device as an Access Token with
3980 "Bearer" token type as defined in IETF RFC 6750. This Access Token is the permission for
3981 the Device to access the OCF Cloud. Received "refreshtoken" is to be treated by Device as a
3982 Refresh Token as defined in IETF RFC 6749. Received "refreshtoken" may be the new
3983 Refresh Token or the same one as provided by the Device in the UPDATE request. In case
3984 when new distinct "refreshtoken" is provided by the OCF Cloud, the Device shall discard the
3985 old value. The OCF Cloud's response values "refreshtoken", "acesstoken" and "expiresin" are
3986 securely stored on the Device.

3987 "oic.r.tokenrefresh" Resource is defined in Table 72.

3988 **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

3989 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)

3991 **13.14 Security Virtual Resources (SVRs) and Access Policy**

3992 The SVRs expose the security-related Properties of the Device.

3993 Granting access requests (RETRIEVE, UPDATE, DELETE, etc.) for these SVRs to
3994 unauthenticated (anonymous) Clients could create privacy or security concerns.

3995 For example, when the Device onboarding State is RFOTM, it is necessary to grant requests for
3996 the oic.r.doxm Resource to anonymous requesters, so that the Device can be discovered and
3997 onboarded by an OBT. Subsequently, it might be preferable to deny requests for the oic.r.doxm
3998 Resource to anonymous requesters, to preserve privacy.

3999 **13.15 SVRs, Discoverability and OCF Endpoints**

4000 All implemented SVRs shall be “discoverable” (reference ISO/IEC 30118-1:2018, Policy
4001 Parameter clause 7.8.2.1.2).

4002 All implemented discoverable SVRs shall expose a Secure OCF Endpoint (e.g. CoAPS)
4003 (reference ISO/IEC 30118-1:2018, clause 10).

4004 The /oic/sec/doxm Resource shall expose an Unsecure OCF Endpoint (e.g. CoAP) in RFOTM
4005 (reference ISO/IEC 30118-1:2018, clause 10).

4006 **13.16 Additional Privacy Consideration for Core and SVRs Resources**

4007 **13.16.1 Additional Privacy Considerations for Core and SVR Resources General**

4008 Unique identifiers are a privacy consideration due to their potential for being used as a tracking
4009 mechanism. These include the following Resources and Properties:

- 4010 – /oic/d Resource containing the ‘di’ and ‘piid’ Properties.
- 4011 – /oic/p Resource containing the ‘pi’ Property.
- 4012 – /oic/sec/doxm Resource containing the ‘deviceuuid’ Property.

4013 All identifiers are unique values that are visible to throughout the Device lifecycle by anonymous
4014 requestors. This implies any Client Device, including those with malicious intent, are able to
4015 reliably obtain identifiers useful for building a log of activity correlated with a specific Platform
4016 and Device.

4017 There are two strategies for privacy protection of Devices:

- 4018 1) Apply an ACL policy that restricts read access to Resources containing unique identifiers
4019 2) Limit identifier persistence to make it impractical for tracking use.
4020 Both techniques can be used effectively together to limit exposure to privacy attacks.

- 4021 1) A Platform / Device manufacturer should specify a default ACL policy that restricts
4022 anonymous requestors from accessing unique identifiers. An OCF Security Domain owner
4023 should modify the ACL policy to grant access to authenticated Devices who, presumably, do
4024 not present a privacy threat.
- 4025 2) Servers shall expose a temporary, non-repeated identifier via an OCF Interface when the
4026 Device transitions to the RESET Device state. The temporary identifiers are disjoint from and
4027 not correlated to the persistent and semi-persistent identifiers. Temporary, non-repeated
4028 identifiers shall be:
- 4029 a) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
 - 4030 b) Generated by a function that is pre-image resistant, second pre-image resistant and
4031 collision resistant

4032 A new Device seeking deployment needs to inform would-be DOTS providers of the identifier
4033 used to begin the onboarding process. However, attackers could obtain the value too and use it
4034 for Device tracking throughout the Device's lifetime.

4035 To address this privacy threat, Servers shall expose a temporary non-repeated identifier via the
4036 deviceuuid Property of the /oic/sec/doxm Resource to unauthenticated /oic/res and /oic/sec/doxm
4037 Resource RETRIEVE requests when the devowneruuid Property of /oic/sec/doxm Resource is the
4038 nil-UUID. The Server shall expose a new temporary non-repeated deviceuuid Property of the
4039 /oic/sec/doxm Resource when the device state transitions to RESET. This ensures the deviceuuid
4040 Property of the /oic/sec/doxm cannot be used to track across multiple owners.

4041 The devowneruuid Property of /oic/sec/doxm Resource is initialized to the nil-UUID upon entering
4042 RESET; which is retained until being set to a non-nil-UUID value during RFOTM device state.
4043 The device shall supply a temporary, non-repeated deviceuuid Property of /oic/sec/doxm
4044 Resource to RETRIEVE requests on /oic/sec/doxm and /oic/res Resources while devowneruuid
4045 Property of /oic/sec/doxm Resource is the nil-UUID. During the OTM process the DOTS shall
4046 UPDATE devowneruuid Property of the /oic/sec/doxm Resource to a non-nil UUID value which is
4047 the trigger for the Device to expose its persistent or semi-persistent device identifier. Therefore
4048 the Device shall supply deviceuuid Property of /oic/sec/doxm Resource in response to RETRIEVE
4049 requests while the devowneruuid Property of the /oic/sec/doxm Resource is a non nil-UUID value.

4050 The DOTS or AMS may also provision an ACL policy that restricts access to the /oic/sec/doxm
4051 Resource such that only authenticated Clients are able to obtain the persistent or semi-persistent
4052 device identifier via the deviceuuid Property value of the /oic/sec/doxm Resource.

4053 Clients avoid making unauthenticated discovery requests that would otherwise reveal a persistent
4054 or semi-persistent identifier using the /oic/sec/cred Resource to first establish an authenticated
4055 connection. This is achieved by first provisioning a /oic/sec/cred Resource entry that contains the
4056 Server's deviceuuid Property value of the /oic/sec/doxm Resource.

4057 The di Property in the /oic/d Resource shall mirror that of the deviceuuid Property of the
4058 /oic/sec/doxm Resource. The DOTS should provision an ACL policy that restricts access to the
4059 /oic/d resource such that only authenticated Clients are able to obtain the di Property of /oic/d
4060 Resource. See Clause 13.1 for deviceuuid Property lifecycle requirements.

4061 Servers should expose a temporary, non-repeated, piid Property of /oic/p Resource Value upon
4062 entering RESET Device state. Servers shall expose a persistent value via the piid Property of
4063 /oic/p Property when the DOTS sets devowneruuid Property to a non-nil-UUID value. An ACL

4064 policy on the /oic/d Resource should protect the piid Property of /oic/p Resource from being
 4065 disclosed to unauthenticated requestors.

4066 Servers shall expose a temporary, non-repeated, pi Property value upon entering RESET Device
 4067 state. Servers shall expose a persistent or semi-persistent platform identifier value via the pi
 4068 Property of the /oic/p Resource when onboarding sets devowneruuid Property to a non-nil-UUID
 4069 value. An ACL policy on the /oic/p Resource should protect the pi Property from being disclosed
 4070 to unauthenticated requestors.

4071 Table 74 depicts Core Resource Properties Access Modes given various Device States.

4072 **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.

4073 Four identifiers are thought to be privacy sensitive:

- 4074 – /oic/d Resource containing the ‘di’ and ‘piid’ Properties.
- 4075 – /oic/p Resource containing the ‘pi’ Property.
- 4076 – /oic/sec/doxm Resource containing the ‘deviceuuid’ Property.

4077 There are three strategies for privacy protection of Devices:

- 4078 1) Apply access control to restrict read access to Resources containing unique identifiers. This
 4079 ensures privacy sensitive identifiers do not leave the Device.
- 4080 2) Limit identifier persistence to make it impractical for tracking use. This ensures privacy
 4081 sensitive identifiers are less effective for tracking and correlation.
- 4082 3) Confidentiality protect the identifiers. This ensures only those authorized to see the value can
 4083 do so.

4084 These techniques can be used to limit exposure to privacy attacks. For example:

- 4085 – ACL policies that restrict anonymous requestors from accessing persistent / semi-persistent
 4086 identifiers can be created.
- 4087 – A temporary identifier can be used instead of a persistent or semi-persistent identifier to
 4088 facilitate onboarding.
- 4089 – Persistent and semi-persistent identifiers can be encrypted before sending them to another
 4090 Device.

4091 A temporary, non-repeated identifier shall be:

- 4092 1) Disjoint from (i.e. not linked to) the persistent or semi-persistent identifiers
4093 2) Generated by a function that is pre-image resistant, second pre-image resistant and collision
4094 resistant

4095 NOTE This requirement is met through a vendor attestation certification mechanism.

4096 **13.16.2 Privacy Protecting the Device Identifiers**

4097 The "di" Property Value of the /oic/d Resource shall mirror that of the "deviceuuid" Property of the
4098 /oic/sec/doxm Resource. The Device should use a new, temporary non-repeated identifier in
4099 place of the "deviceuuid" Property Value of /oic/sec/doxm Resource upon entering the RESET
4100 Device state. This value should be exposed while the "devowneruuid" Property has a nil UUID
4101 value. The Device should expose its persistent (or semi-persistent) "deviceuuid" Property value
4102 of the /oic/sec/doxm Resource after the DOTS sets the "devowneruuid" Property to a non-nil-
4103 UUID value. The temporary identifier should not change more frequently than once per Device
4104 state transition to RESET.

4105 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4106 – If constructing a CRUDN response for any Resource that contains the "deviceuuid" and/or "di"
4107 Property values:
 - 4108 – The Device should include its persistent (or semi-persistent) "deviceuuid" (or "di") Property
4109 value only if responding to an authenticated requestor and the "deviceuuid" (or "di") value
4110 is confidentiality protected .
 - 4111 – The Device should use a temporary non-repeated "deviceuuid" (or "di") Property value if
4112 responding to an unauthenticated requestor.
- 4113 – The AMS should provision an ACL policy on the /oic/sec/doxm and /oic/d resources to further
4114 protect the "deviceuuid" and "di" Properties from being disclosed unnecessarily.

4115 See 13.2 for deviceuuid Property lifecycle requirements.

4116 NOTE A Client Device can avoid disclosing its persistent (or semi-persistent) identifiers by avoiding unnecessary
4117 discovery requests. This is achieved by provisioning a /oic/sec/cred Resource entry that contains the Server's
4118 deviceuuid Property value. The Client establishes a secure connection to the Server straight away.

4119 **13.16.3 Privacy Protecting the Protocol Independent Device Identifier**

4120 The Device should use a new, temporary non-repeated identifier in place of the "piid" Property
4121 Value of /oic/d Resource upon entering the RESET Device state. If a temporary, non-repeated
4122 value has been generated, it should be used while the "devowneruuid" Property has the nil UUID
4123 value. The Device should use its persistent "piid" Property value after the DOTS sets the
4124 "devowneruuid" Property to a non-nil-UUID value. The temporary identifier should not change
4125 more frequently than once per Device state transition to RESET.

4126 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4127 – If constructing a CRUDN response for any Resource that contains the "piid" Property value:
 - 4128 – The Device should include its persistent "piid" Property value only if responding to an
4129 authenticated requestor and the "piid" value is confidentiality protected.
 - 4130 – The Device should include a temporary non-repeated "piid" Property value if responding to
4131 an unauthenticated requestor.
- 4132 – The AMS should provision an ACL policy on the /oic/d Resource to further protect the piid
4133 Property of /oic/p Resource from being disclosed unnecessarily.

4134 **13.16.4 Privacy Protecting the Platform Identifier**

4135 The Device should use a new, temporary non-repeated identifier in place of the "pi" Property
4136 Value of the /oic/p Resource upon entering the RESET Device state. This value should be
4137 exposed while the "devowneruuid" Property has a nil UUID value. The Device should use its

4138 persistent (or semi-persistent) "pi" Property value after the DOTS sets the "devowneruuid"
 4139 Property to a non-nil-UUID value. The temporary identifier should not change more frequently
 4140 than once per Device state transition to RESET.

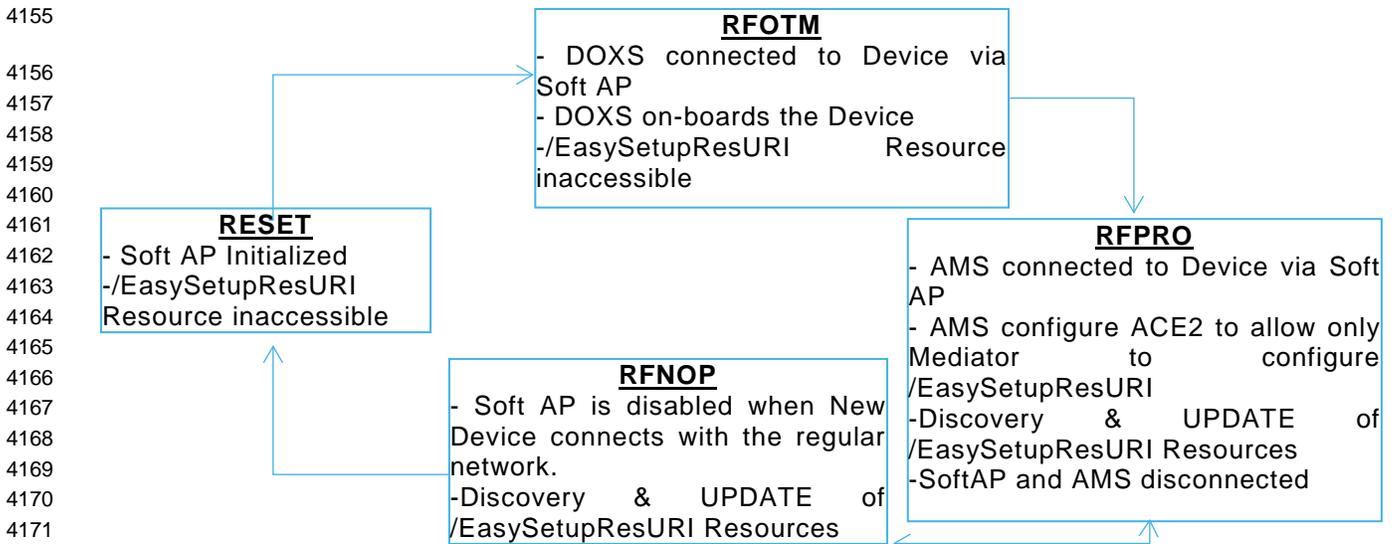
4141 Subsequent to the "devowneruuid" being UPDATED to a non-nil UUID:

- 4142 – If constructing a CRUDN response for any Resource that contains the "pi" Property value:
 - 4143 – The Device should include its persistent (or semi-persistent) "pi" Property value only if
 - 4144 responding to an authenticated requestor and the "pi" value is confidentiality protected.
 - 4145 – The Device should include a temporary non-repeated "pi" Property value if responding to
 - 4146 an unauthenticated requestor.
- 4147 – The AMS should provision an ACL policy on the /oic/p Resource to protect the pi Property
- 4148 from being disclosed unnecessarily.

4149 **13.17 Easy Setup Resource Device State**

4150 This clause only applies to New Device that uses Easy Setup for Ownership Transfer as defined
 4151 in ISO/IEC 30118-7:2018. Easy setup has no impact to New Devices that have a different way of
 4152 connecting to the network i.e. DOTS and AMS don't use a Soft AP to connect to non-Easy Setup
 4153 Devices.

4154 Figure 39 shows an example of Soft AP and Easy Setup Resource in different Device states.



4172 **Figure 39 – Example of Soft AP and Easy Setup Resource in different Device states**

4173 Device enters RFOTM Device state, Soft AP may be accessible in RFOTM and RFPRO Device's
 4174 state.

4175 While it is reasonable for a user to expect that power cycling a New Device will turn on the Soft
 4176 AP for Easy Setup during the initial setup, since that is potentially how it behaved on first boot, it
 4177 is a security risk to make this the default behavior of a device that remains unenrolled beyond a
 4178 reasonable period after first boot.

4179 Therefore, the Soft AP for Easy Setup has several requirements to improve security:

- 4180 – Time availability of Easy Setup Soft AP should be minimised, and shall not exceed 30 minutes
- 4181 after Device factory reset RESET or first power boot, or when user initiates the Soft AP for
- 4182 Easy Setup.
- 4183 – If a New Device tried and failed to complete Easy Setup Enrollment immediately following the
- 4184 first boot, or after a factory reset, it may turn the Easy Setup Soft AP back on automatically

4185 for another 30 mins upon being power cycled, provided that the power cycle occurs within 3
4186 hours of first boot or the most recent factory reset. If the user has initiated the Easy Setup
4187 Soft AP directly without a factory reset, it is not necessary to turn it back on if it was on
4188 immediately prior to power cycle, because the user obviously knows how to initiate the
4189 process manually.

- 4190 – After 3 hours from first boot or factory reset without successfully enrolling the device, the Soft
4191 AP should not turn back on for Easy Setup until another factory reset occurs, or the user
4192 initiates the Easy Setup Soft AP directly.
- 4193 – Easy Setup Soft AP may stay enabled during RFNOP, until the Mediator instructs the New
4194 Device to connect to the Enroller.
- 4195 – The Easy Setup Soft AP shall be disabled when the New Device successfully connects to the
4196 Enroller.
- 4197 – Once a New Device has successfully connected to the Enroller, it shall not turn the Easy
4198 Setup Soft AP back on for Easy Setup Enrollment again unless the Device is factory reset, or
4199 the user initiates the Easy Setup Soft AP directly.
- 4200 – Just Works OTM shall not be enabled on Devices which support Easy Setup.
- 4201 – The Soft AP shall be secured (e.g. shall not expose an open AP).
- 4202 – The Soft AP shall support a passphrase for connection by the Mediator, and the passphrase
4203 shall be between and 8 and 64 ASCII printable characters. The passphrase may be printed
4204 on a label, sticker, packaging etc., and may be entered by the user into the Mediator device.
- 4205 – The Soft AP should not use a common passphrase across multiple Devices. Instead, the
4206 passphrase may be sufficiently unique per device, to prevent guessing of the passphrase by
4207 an attacker with knowledge of the Device type, model, manufacturer, or any other information
4208 discoverable through Device's exposed interfaces.

4209 The Enrollee shall support WPA2 security (i.e. shall list WPA2 in the "swat" Property of the
4210 /example/WiFiConfResURI Resource), for potential selection by the Mediator in connecting the
4211 Enrollee to the Enroller. The Mediator should select the best security available on the Enroller,
4212 for use in connecting the Enrollee to the Enroller.

4213 The Enrollee may not expose any interfaces (e.g. web server, debug port, NCRs, etc.) over the
4214 Soft AP, other than SVRs, and Resources required for Wi-Fi Easy Setup.

4215 The /example/EasySetupResURI Resource should not be discoverable in RFOTM or SRESET
4216 state. After Ownership Transfer process is completed with the DOTS, and the Device enters in
4217 RFPRO Device state, the /example/EasySetupResURI may be Discoverable. The DOTS may be
4218 hosted on the Mediator Device.

4219 The OTM CoAPS session may be used by Mediator for connection over Soft AP for ownership
4220 transfer and initial Easy Setup provisioning. SoftAP or regular network connection may be used
4221 by AMS for /oic/sec/acl2 Resource provisioning in RFPRO state. The CoAPS session
4222 authentication and encryption is already defined in the Security spec.

4223 In RFPRO state, AMS should configure ACL2 Resource on the Device with ACE2 for following
4224 Resources to be only configurable by the Mediator Device with permission to UPDATE or
4225 RETRIEVE access:

- 4226 – /example/EasySetupResURI
- 4227 – /example/WifiConfResURI
- 4228 – /example/DevConfResURI

4229 An ACE2 granting RETRIEVE or UPDATE access to the Easy Setup Resource

4230 {

```
4231     "subject": { "uuid": "<insert-UUID-of-Mediator>" },
4232     "resources": [
4233         { "href": "/example/EasySetupResURI" },
4234         { "href": "/example/WiFiConfResURI" },
4235         { "href": "/example/DevConfResURI" },
4236     ],
4237     "permission": 6 // RETRIEVE (2) or UPDATE and RETRIEVE(6)
4238 }
```

4239 ACE2 may be re-configured after Easy Setup process. These ACE2s should be installed prior to
4240 the Mediator performing any RETRIEVE/UPDATE operations on these Resources.

4241 In RFPRO or RFNOP, the Mediator should discover /EasySetupResURI Resources and UPDATE
4242 these Resources. The AMS may UPDATE /EasySetupResURI resources in RFNOP Device state.

4243 **14 Security Hardening Guidelines/ Execution Environment Security**

4244 **14.1 Preamble**

4245 This is an informative clause. Many TGs in OCF have security considerations for their protocols
4246 and environments. These security considerations are addressed through security mechanisms
4247 specified in the security documents for OCF. However, effectiveness of these mechanisms
4248 depends on security robustness of the underlying hardware and software Platform. This clause
4249 defines the components required for execution environment security.

4250 **14.2 Execution Environment Elements**

4251 **14.2.1 Execution Environment Elements General**

4252 Execution environment within a computing Device has many components. To perform security
4253 functions in a robustness manner, each of these components has to be secured as a separate
4254 dimension. For instance, an execution environment performing AES cannot be considered secure
4255 if the input path entering keys into the execution engine is not secured, even though the
4256 partitions of the CPU, performing the AES encryption, operate in isolation from other processes.
4257 Different dimensions referred to as elements of the execution environment are listed below. To
4258 qualify as a secure execution environment (SEE), the corresponding SEE element must qualify as
4259 secure.

- 4260 – (Secure) Storage
- 4261 – (Secure) Execution engine
- 4262 – (Trusted) Input/output paths
- 4263 – (Secure) Time Source/clock
- 4264 – (Random) number generator
- 4265 – (Approved) cryptographic algorithms
- 4266 – Hardware Tamper (protection)

4267 **NOTE** Software security practices (such as those covered by OWASP) are outside scope of this document, as
4268 development of secure code is a practice to be followed by the open source development community. This document
4269 will however address the underlying Platform assistance required for executing software. Examples are secure boot
4270 and secure software upgrade.

4271 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,
4272 14.2.7.

4273 **14.2.2 Secure Storage**

4274 **14.2.2.1 Secure Storage General**

4275 Secure storage refers to the physical method of housing sensitive or confidential data ("Sensitive
4276 Data"). Such data could include but not be limited to symmetric or asymmetric private keys,
4277 certificate data, OCF Security Domain access credentials, or personal user information. Sensitive
4278 Data requires that its integrity be maintained, whereas *Critical* Sensitive Data requires that both
4279 its integrity and confidentiality be maintained.

4280 It is strongly recommended that IoT Device makers provide reasonable protection for Sensitive
4281 Data so that it cannot be accessed by unauthorized Devices, groups or individuals for either
4282 malicious or benign purposes. In addition, since Sensitive Data is often used for authentication
4283 and encryption, it must maintain its integrity against intentional or accidental alteration.

4284 A partial list of Sensitive Data is outlined in Table 75:

Table 75 – Examples of Sensitive Data

Data	Integrity protection	Confidentiality protection
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

4286 Exact method of protection for secure storage is implementation specific, but typically
4287 combinations of hardware and software methods are used.

4288 **14.2.2.2 Hardware Secure Storage**

4289 Hardware secure storage is recommended for use with critical Sensitive Data such as symmetric
4290 and asymmetric private keys, access credentials, and personal private data. Hardware secure
4291 storage most often involves semiconductor-based non-volatile memory ("NVRAM") and includes
4292 countermeasures for protecting against unauthorized access to Critical Sensitive Data.

4293 Hardware-based secure storage not only stores Sensitive Data in NVRAM, but also provides
4294 protection mechanisms to prevent the retrieval of Sensitive Data through physical and/or
4295 electronic attacks. It is not necessary to prevent the attacks themselves, but an attempted attack
4296 should not result in an unauthorized entity successfully retrieving Sensitive Data.

4297 Protection mechanisms should provide JIL Moderate protection against access to Sensitive Data
4298 from attacks that include but are not limited to:

- 4299 1) Physical decapping of chip packages to optically read NVRAM contents
- 4300 2) Physical probing of decapped chip packages to electronically read NVRAM contents
- 4301 3) Probing of power lines or RF emissions to monitor voltage fluctuations to discern the bit
4302 patterns of Critical Sensitive Data
- 4303 4) Use of malicious software or firmware to read memory contents at rest or in transit within a
4304 microcontroller
- 4305 5) Injection of faults that induce improper Device operation or loss or alteration of Sensitive Data

4306 **14.2.2.3 Software Storage**

4307 It is generally NOT recommended to rely solely on software and unsecured memory to store
4308 Sensitive Data even if it is encrypted. Critical Sensitive Data such as authentication and
4309 encryption keys should be housed in hardware secure storage whenever possible.

4310 Sensitive Data stored in volatile and non-volatile memory shall be encrypted using acceptable
4311 algorithms to prevent access by unauthorized parties through methods described in 14.2.2.2.

4312 **14.2.2.4 Additional Security Guidelines and Best Practices**

4313 Some general practices that can help ensure that Sensitive Data is not compromised by various
4314 forms of security attacks:

- 4315 1) FIPS Random Number Generator ("RNG") – Insufficient randomness or entropy in the RNG
4316 used for authentication challenges can substantially degrade security strength. For this
4317 reason, it is recommended that a FIPS 800-90A-compliant RNG with a certified noise source
4318 be used for all authentication challenges.
- 4319 2) Secure download and boot – To prevent the loading and execution of malicious software,
4320 where it is practical, it is recommended that Secure Download and Secure Boot methods that
4321 authenticate a binary's source as well as its contents be used.
- 4322 3) Deprecated algorithms – Algorithms included but not limited to the list below are considered
4323 insecure and shall not be used for any security-related function:
 - 4324 a) SHA-1
 - 4325 b) MD5
 - 4326 c) RC4
 - 4327 d) RSA 1024
- 4328 4) Encrypted transmission between blocks or components – Even if critical Sensitive Data is
4329 stored in Secure Storage, any use of that data that requires its transmission out of that
4330 Secure Storage should be encrypted to prevent eavesdropping by malicious software within
4331 an MCU/MPU.
- 4332 5) It is recommended to avoid using wildcard in Subject Id ("*"), when setting up oic.r.cred
4333 Resource entries, since this opens up an identity spoofing opportunity.
- 4334 6) Device vendor understands that it is the Device vendor's responsibility to ensure the Device
4335 meets security requirements for its intended uses. As an example, IoTivity is a reference
4336 implementation intended to be used as a basis for a product, but IoTivity has not undergone
4337 3rd party security review, penetration testing, etc. Any Device based on IoTivity should
4338 undergo appropriate penetration testing and security review prior to sale or deployment.
- 4339 7) Device vendor agrees to publish the expected support lifetime for the Device to OCF and to
4340 consumers. Changes should be made to a public and accessible website. Expectations
4341 should be clear as to what will be supported and for how long the Device vendor expects to
4342 support security updates to the software, operating system, drivers, networking, firmware and
4343 hardware of the device.
- 4344 8) Device vendor has not implemented test or debug interfaces on the Device which are
4345 operable or which can be enabled which might present an attack vector on the Device which
4346 circumvents the interface-level security or access policies of the Device.
- 4347 9) Device vendor understands that if an application running on the Device has access to
4348 cryptographic elements such as the private keys or Ownership Credential, then those
4349 elements have become vulnerable. If the Device vendor is implementing a Bridge, an OBT, or
4350 a Device with access to the Internet beyond the local network, the execution of critical
4351 functions should take place within a Trusted or Secure Execution Environment (TEE/SEE).
- 4352 10) Any PINs or fixed passphrases used for onboarding, WiFi Easy Setup, SoftAP management
4353 or access, or other security-critical function, should be sufficiently unique (do not duplicate
4354 passphrases. The creation of these passphrases or PINS should not be algorithmically
4355 deterministic nor should they use insufficient entropy in their creation.
- 4356 11) Ensure that there are no remaining "VENDOR_TODO" items in the source code.

4357 12) If the implementation of this document uses the "Just Works" onboarding method, understand
4358 that there is a man-in-the-middle vulnerability during the onboarding process where a
4359 malicious party could intercept messages between the device being onboarded and the OBT
4360 and could persist, acting as an intermediary with access to message traffic, during the lifetime
4361 of that onboarded device. The recommended best practice would be to use an alternate
4362 ownership transfer method (OTM) instead of "Just Works".

4363 13) It is recommended that at least one static and dynamic analysis tool¹ be applied to any
4364 proposed major production release of the software before its release, and any vulnerabilities
4365 resolved.

4366 14) To avoid a malicious device being able to covertly join an OCF Security Domain,
4367 implementers of any onboarding tool may eliminate completely autonomous sequences where
4368 a device is brought into the OCF Security Domain without any authorization by the owner.
4369 Consider either including a confirmation with the OCF Security Domain owner/operator (e.g.
4370 "Do you want to add 'LIGHTBULB 80' from manufacturer 'GenericLightingCo'?
4371 Yes/No/Cancel?") or a confirmation with a security policy (e.g. an enterprise policy where the
4372 OCF Security Domain admin can bulk-onboard devices).

4373 **14.2.3 Secure execution engine**

4374 Execution engine is the part of computing Platform that processes security functions, such as
4375 cryptographic algorithms or security protocols (e.g. DTLS). Securing the execution engine
4376 requires the following

4377 – Isolation of execution of sensitive processes from unauthorized parties/ processes. This
4378 includes isolation of CPU caches, and all of execution elements that needed to be considered
4379 as part of trusted (crypto) boundary.

4380 – Isolation of data paths into and out of execution engine. For instance both unencrypted but
4381 sensitive data prior to encryption or after decryption, or cryptographic keys used for
4382 cryptographic algorithms, such as decryption or signing. See clause 14.2.4 for more details.

4383 **14.2.4 Trusted input/output paths**

4384 Paths/ ports used for data entry into or export out of trusted/ crypto-boundary needs to be
4385 protected. This includes paths into and out secure execution engine and secure memory.

4386 Path protection can be both hardware based (e.g. use of a privileged bus) or software based
4387 (using encryption over an untrusted bus).

4388 **14.2.5 Secure clock**

4389 Many security functions depend on time-sensitive credentials. Examples are time stamped
4390 Kerberos tickets, OAUTH tokens, X.509 certificates, OSCP response, software upgrades, etc.
4391 Lack of secure source of clock can mean an attacker can modify the system clock and fool the
4392 validation mechanism. Thus an SEE needs to provide a secure source of time that is protected
4393 from tampering. Trustworthiness from security robustness standpoint is not the same as accuracy.
4394 Protocols such as NTP can provide rather accurate time sources from the network, but are not
4395 immune to attacks. A secure time source on the other hand can be off by seconds or minutes
4396 depending on the time-sensitivity of the corresponding security mechanism. Secure time source
4397 can be external as long as it is signed by a trusted source and the signature validation in the
4398 local Device is a trusted process (e.g. backed by secure boot).

4399 **14.2.6 Approved algorithms**

4400 An important aspect of security of the entire ecosystem is the robustness of publicly vetted and
4401 peer-reviewed (e.g. NIST-approved) cryptographic algorithms. Security is not achieved by
4402 obscurity of the cryptographic algorithm. To ensure both interoperability and security, not only

¹ A general discussion of analysis tools can be found here: <https://www.ibm.com/developerworks/library/se-static/>

4403 widely accepted cryptographic algorithms must be used, but also a list of approved cryptographic
4404 functions must be specified explicitly. As new algorithms are NIST approved or old algorithms are
4405 deprecated, the list of approved algorithms must be maintained by OCF. All other algorithms
4406 (even if they deemed stronger by some parties) must be considered non-approved.

4407 The set of algorithms to be considered for approval are algorithms for

- 4408 – Hash functions
- 4409 – Signature algorithms
- 4410 – Encryption algorithms
- 4411 – Key exchange algorithms
- 4412 – Pseudo Random functions (PRF) used for key derivation

4413 This list will be included in this or a separate security robustness rules document and must be
4414 followed for all security specifications within OCF.

4415 **14.2.7 Hardware tamper protection**

4416 Various levels of hardware tamper protection exist. We borrow FIPS 140-2 terminology (not
4417 requirements) regarding tamper protection for cryptographic module

- 4418 – Production-grade (lowest level): this means components that include conformal sealing
4419 coating applied over the module's circuitry to protect against environmental or other physical
4420 damage. This does not however require zeroization of secret material during physical
4421 maintenance. This definition is borrowed from FIPS 140-2 security level 1.
- 4422 – Tamper evident/proof (mid-level), This means the Device shows evidence (through covers,
4423 enclosures, or seals) of an attempted physical tampering. This definition is borrowed from
4424 FIPS 140-2 security level 2.
- 4425 – Tamper resistance (highest level), this means there is a response to physical tempering that
4426 typically includes zeroization of sensitive material on the module. This definition is borrowed
4427 from FIPS 140-2 security level 3.

4428 It is difficult of specify quantitative certification test cases for accreditation of these levels.
4429 Content protection regimes usually talk about different tools (widely available, specialized and
4430 professional tools) used to circumvent the hardware protections put in place by manufacturing. If
4431 needed, OCF can follow that model, if and when OCF engage in distributing sensitive key
4432 material (e.g. PKI) to its members.

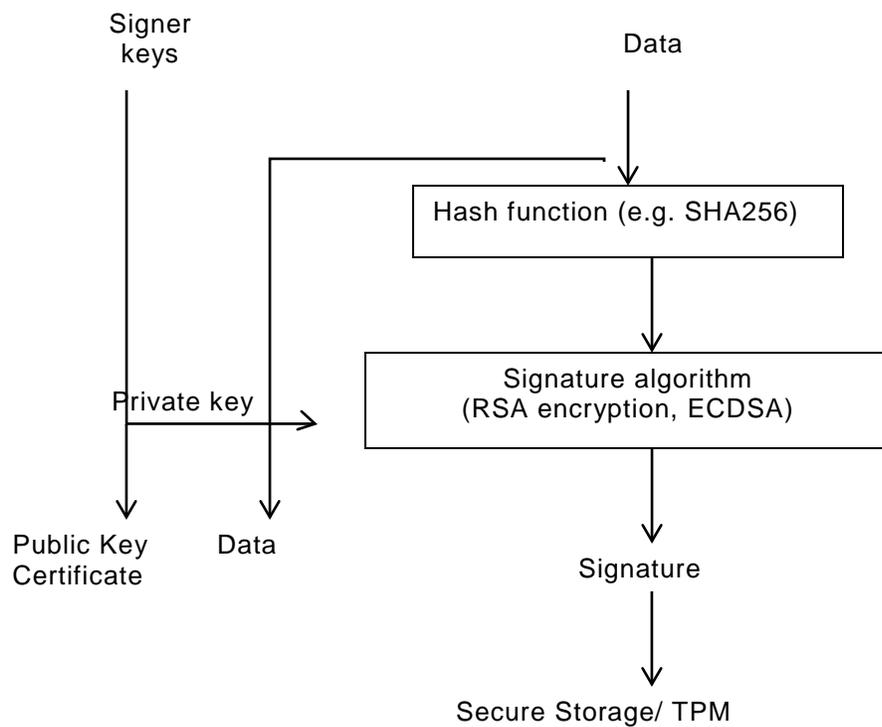
4433 **14.3 Secure Boot**

4434 **14.3.1 Concept of software module authentication**

4435 In order to ensure that all components of a Device are operating properly and have not been
4436 tampered with, it is best to ensure that the Device is booted properly. There may be multiple
4437 stages of boot. The end result is an application running on top an operating system that takes
4438 advantage of memory, CPU and peripherals through drivers.

4439 The general concept is the each software module is invoked only after cryptographic integrity
4440 verification is complete. The integrity verification relies on the software module having been
4441 hashed (e.g. SHA_1, SHA_256) and then signed with a cryptographic signature algorithm with
4442 (e.g. RSA), with a key that only a signing authority has access to.

4443 Figure 40 depicts software module authentication.

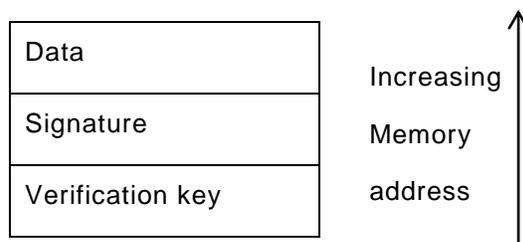


4444

Figure 40 – Software Module Authentication

4445 After the data is signed with the signer’s signing key (a private key), the verification key (the
 4446 public key corresponding to the private signing key) is provided for later verification. For lower
 4447 level software modules, such as bootloaders, the signatures and verification keys are inserted
 4448 inside tamper proof memory, such as one-time programmable memory or TPM. For higher level
 4449 software modules, such as application software, the signing is typically performed according to
 4450 the PKCS#7 format IETF RFC 2315, where the signedData format includes both indications for
 4451 signature algorithm, hash algorithm as well as the signature verification key (or certificate).
 4452 Secure boot does not require use of PKCS#7 format.

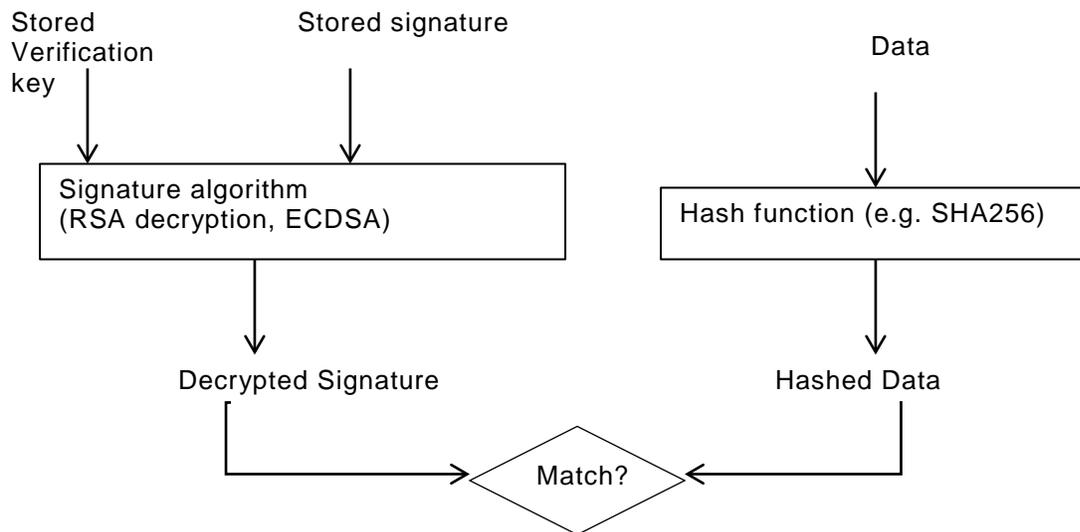
4453 Figure 41 depicts verification software module.



4454

Figure 41 – Verification Software Module

4455 As shown in Figure 42. the verification module first decrypts the signature with the verification
 4456 key (public key of the signer). The verification module also calculates a hash of the data and then
 4457 compares the decrypted signature (the original) with the hash of data (actual) and if the two
 4458 values match, the software module is authentic.



4459 **Figure 42 – Software Module Authenticity**

4459

4460 **14.3.2 Secure Boot process**

4461 Depending on the Device implementation, there may be several boot stages. Typically, in a PC/
 4462 Linux type environment, the first step is to find and run the BIOS code (first-stage bootloader) to
 4463 find out where the boot code is and then run the boot code (second-stage boot loader). The
 4464 second stage bootloader is typically the process that loads the operating system (Kernel) and
 4465 transfers the execution to the where the Kernel code is. Once the Kernel starts, it may load
 4466 external Kernel modules and drivers.

4467 When performing a secure boot, it is required that the integrity of each boot loader is verified
 4468 before executing the boot loader stage. As mentioned, while the signature and verification key for
 4469 the lowest level bootloader is typically stored in tamper-proof memory, the signature and
 4470 verification key for higher levels should be embedded (but attached in an easily accessible
 4471 manner) in the data structures software.

4472 **14.3.3 Robustness Requirements**

4473 **14.3.3.1 Robustness General**

4474 To qualify as high robustness secure boot process, the signature and hash algorithms shall be
 4475 one of the approved algorithms, the signature values and the keys used for verification shall be
 4476 stored in secure storage and the algorithms shall run inside a secure execution environment and
 4477 the keys shall be provided the SEE over trusted path.

4478 **14.3.3.2 Next steps**

4479 Develop a list of approved algorithms and data formats

4480 **14.4 Attestation**

4481 **14.5 Software Update**

4482 **14.5.1 Overview:**

4483 The Device lifecycle does not end at the point when a Device is shipped from the manufacturer;
 4484 the distribution, retailing, purchase, installation/onboarding, regular operation, maintenance and
 4485 end-of-life stages for the Device remain outstanding. It is possible for the Device to require

4486 update during any of these stages, although the most likely times are during onboarding, regular
4487 operation and maintenance. The aspects of the software include, but are not limited to, firmware,
4488 operating system, networking stack, application code, drivers, etc.

4489 **14.5.2 Recognition of Current Differences**

4490 Different manufacturers approach software update utilizing a collection of tools and strategies:
4491 over-the-air or wired USB connections, full or partial replacement of existing software, signed and
4492 verified code, attestation of the delivery package, verification of the source of the code, package
4493 structures for the software, etc.

4494 It is recommended that manufacturers review their processes and technologies for compliance
4495 with industry best-practices that a thorough security review of these takes place and that periodic
4496 review continue after the initial architecture has been established.

4497 This document applies to software updates as recommended to be implemented by Devices; it
4498 does not have any bearing on the above-mentioned alternative proprietary software update
4499 mechanisms.

4500 **14.5.3 Software Version Validation**

4501 Setting the Initiate Software Version Validation bit in the `/oic/sec/pstat.tm` Property (see Table 57
4502 defines the Properties of "oic.r.pstat".

4503 Table 57 of 13.8) indicates a request to initiate the software version validation process, the
4504 process whereby the Device validates the software (including firmware, operating system, Device
4505 drivers, networking stack, etc.) against a trusted source to see if, at the conclusion of the check,
4506 the software update process will need to be triggered (see clause 14.5.4). When the Initiate
4507 Software Version Validation bit of `/oic/sec/pstat.tm` is set to 1 (TRUE) by a sufficiently privileged
4508 Client, the Device sets the `/oic/sec/pstat.cm` Initiate Software Version Validation bit to 0 and
4509 initiates a software version check. Once the Device has determined if an update is available, it
4510 sets the Initiate Software Version Validation bit in the `/oic/sec/pstat.cm` Property to 1 (TRUE) if an
4511 update is available or 0 (FALSE) if no update is available. To signal completion of the Software
4512 Version Validation process, the Device sets the Initiate Software Version Validation bit in the
4513 `/oic/sec/pstat.tm` Property back to 0 (FALSE). If the Initiate Software Version Validation bit of
4514 `/oic/sec/pstat.tm` is set to 0 (FALSE) by a Client, it has no effect on the validation process.

4515 **14.5.4 Software Update**

4516 Setting the Initiate Secure Software Update bit in the `/oic/sec/pstat.tm` Property (see Table 57
4517 defines the Properties of "oic.r.pstat".

4518 Table 57 of 13.8) indicates a request to initiate the software update process. When the Initiate
4519 Secure Software Update bit of `/oic/sec/pstat.tm` is set to 1 (TRUE) by a sufficiently privileged
4520 Client, the Device sets the `/oic/sec/pstat.cm` Initiate Software Version Validation bit to 0 and
4521 initiates a software update process. Once the Device has completed the software update process,
4522 it sets the Initiate Secure Software Update bit in the `/oic/sec/pstat.cm` Property to 1 (TRUE)
4523 if/when the software was successfully updated or 0 (FALSE) if no update was performed. To
4524 signal completion of the Secure Software Update process, the Device sets the Initiate Secure
4525 Software Update bit in the `/oic/sec/pstat.tm` Property back to 0 (FALSE). If the Initiate Secure
4526 Software Update bit of `/oic/sec/pstat.tm` is set to 0 (FALSE) by a Client, it has no effect on the
4527 update process.

4528 **14.5.5 Recommended Usage**

4529 The Initiate Secure Software Update bit of `/oic/sec/pstat.tm` should only be set by a Client after
4530 the Initiate Software Version Validation check is complete.

4531 The process of updating Device software may involve state changes that affect the Device
4532 Operational State (/oic/sec/pstat.dos). Devices with an interest in the Device(s) being updated
4533 should monitor /oic/sec/pstat.dos and be prepared for pending software update(s) to affect
4534 Device state(s) prior to completion of the update.

4535 The Device itself may indicate that it is autonomously initiating a software version check/update
4536 or that a check/update is complete by setting the pstat.tm and pstat.cm Initiate Software Version
4537 Validation and Secure Software Update bits when starting or completing the version check or
4538 update process. As is the case with a Client-initiated update, Clients can be notified that an
4539 autonomous version check or software update is pending and/or complete by observing pstat
4540 resource changes.

4541 **14.6 Non-OCF Endpoint interoperability**

4542 **14.7 Security Levels**

4543 Security Levels are a way to differentiate Devices based on their security criteria. This need for
4544 differentiation is based on the requirements from different verticals such as industrial and health
4545 care and may extend into smart home. This differentiation is distinct from Device classification
4546 (e.g. IETF RFC 7228)

4547 These categories of security differentiation may include, but is not limited to:

- 4548 1) Security Hardening
- 4549 2) Identity Attestation
- 4550 3) Certificate/Trust
- 4551 4) Onboarding Technique
- 4552 5) Regulatory Compliance
 - 4553 a) Data at rest
 - 4554 b) Data in transit
- 4555 6) Cipher Suites – Crypto Algorithms & Curves
- 4556 7) Key Length
- 4557 8) Secure Boot/Update

4558 In the future security levels can be used to define interoperability.

4559 The following applies to the OCF Security Specification 1.1:

4560 The current document does not define any other level beyond Security Level 0. All Devices will
4561 be designated as Level 0. Future versions may define additional levels.

4562 Additional comments:

- 4563 – The definition of a given security level will remain unchanged between versions of the
4564 document.
- 4565 – Devices that meet a given level may, or may not, be capable of upgrading to a higher level.
- 4566 – Devices may be evaluated and re-classified at a higher level if it meets the requirements of
4567 the higher level (e.g. if a Device is manufactured under the 1.1 version of the document, and
4568 a later document version defines a security level 1, the Device could be evaluated and
4569 classified as level 1 if it meets level 1 requirements).
- 4570 – The security levels may need to be visible to the end user.

4571 **14.8 Security Profiles**

4572 **14.8.1 Security Profiles General**

4573 Security Profiles are a way to differentiate OCF Devices based on their security criteria. This
4574 need for differentiation is based on the requirements from different verticals such as industrial
4575 and health care and may extend into smart home. This differentiation is distinct from device
4576 classification (e.g. IETF RFC 7228)

4577 These categories of security differentiation may include, but is not limited to:

- 4578 1) Security Hardening and assurances criteria
- 4579 2) Identity Attestation
- 4580 3) Certificate/Trust
- 4581 4) Onboarding Technique
- 4582 5) Regulatory Compliance
 - 4583 a) Data at rest
 - 4584 b) Data in transit
- 4585 6) Cipher Suites – Crypto Algorithms & Curves
- 4586 7) Key Length
- 4587 8) Secure Boot/Update

4588 Each Security Profile definition must specify the version or versions of the OCF Security
4589 Specification(s) that form a baseline set of normative requirements. The profile definition may
4590 include security requirements that supersede baseline requirements (not to relax security
4591 requirements).

4592 Security Profiles have the following properties:

- 4593 – A given profile definition is not specific to the version of the document that defines it. For
4594 example, the profile may remain constant for subsequent OCF Security Specification versions.
- 4595 – A specific OCF Device and platform combination may be used to satisfy the security profile.
- 4596 – Profiles may have overlapping criteria, hence it may be possible to satisfy multiple profiles
4597 simultaneously.
- 4598 – An OCF Device that satisfied a profile initially may be re-evaluated at a later time and found
4599 to satisfy a different profile (e.g. if a device is manufactured under the 1.1 version of the
4600 document, and a later document version defines a security profile Black, the device could be
4601 evaluated and classified as profile Black if it meets profile Black requirements).
- 4602 – A machine-readable representation of compliance results specifically describing profiles
4603 satisfied may be used to facilitate OCF Device onboarding. (e.g. a manufacturer certificate or
4604 manifest may contain security profiles attributes).

4605 **14.8.2 Identification of Security Profiles (Normative)**

4606 **14.8.2.1 Security Profiles in Prior Documents**

4607 OCF Devices conforming to versions of the OCF Security Specifications where Security Profiles
4608 Resource was not defined may be presumed to satisfy the "sp-baseline-v0" profile (defined in
4609 14.8.3.3) or may be regarded as unspecified. If Security Profile is unspecified, the Client may use
4610 the OCF Security Specification version to characterize expected security behavior.

4611 **14.8.2.2 Security Profile Resource Definition**

4612 The oic.sec.sp Resource is used by the OCF Device to show which OCF Security Profiles the
4613 OCF Device is capable of supporting and which are authorized for use by the OCF Security

4614 Domain owner. Properties of the Resource identify which OCF Security Profile is currently
 4615 operational. The ocfSecurityProfileOID value type shall represent OID values and may reference
 4616 an entry in the form of strings (UTF-8).

4617 "oic.sec.sp" Resource is defined in Table 76.

4618 **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

4619 Table 77 defines the Properties of "oic.sec.sp".

4620 **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")

4621 The following OIDs are defined to uniquely identify Security Profiles. Future Security Profiles or
 4622 changes to existing Security Profiles may result in a new ocfSecurityProfileOID.

4623 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
 4624 private(4) enterprise(1) OCF(51414) }

4625
 4626 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }

4627
 4628 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }

4629
 4630 sp-unspecified ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
 4631 --The Security Profile is not specified

4632 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
 4633 --This specifies the OCF Baseline Security Profile(s)

4634 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
 4635 --This specifies the OCF Black Security Profile(s)

4636 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
 4637 --This specified the OCF Blue Security Profile(s)

4638 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
 4639 --This specifies the OCF Purple Security Profile(s)

4640
 4641 --versioned Security Profiles
 4642 sp-unspecified-v0 ::= ocfSecurityProfileOID (id-sp-unspecified 0)
 4643 --v0 of unspecified security profile, "1.3.6.1.4.1.51414.0.0.0.0"

4644 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
 4645 --v0 of baseline security profile, "1.3.6.1.4.1.51414.0.0.1.0"

4646 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
 4647 --v0 of black security profile, "1.3.6.1.4.1.51414.0.0.2.0"

4648 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
 4649 --v0 of blue security profile, "1.3.6.1.4.1.51414.0.0.3.0"

4650 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
 4651 --v0 of purple security profile, "1.3.6.1.4.1.51414.0.0.4.0"

4652
 4653 ocfSecurityProfileOID ::= UTF8String

4654

4655 **14.8.3 Security Profiles**

4656 **14.8.3.1 Security Profiles General**

4657 The Security Profiles Resource shall be pre-populated with manufacturer default values (Refer to
4658 the Security Profile clauses for additional details).

4659 The OCF Conformance criteria may require vendor attestation that establishes the expected
4660 environment in which the OCF Device is hosted (Refer to the Security Profile clauses for specific
4661 requirements).

4662 **14.8.3.2 Security Profile Unspecified (sp-unspecified-v0)**

4663 The Security Profile "sp-unspecified-v0" is reserved for future use.

4664 **14.8.3.3 Security Profile Baseline v0 (sp-baseline-v0)**

4665 The Security Profile "sp-baseline-v0" is defined for all OCF Security Specification versions where
4666 the /oic/sec/sp Resource is defined. All Devices shall include the "sp-baseline-v0" OID in the
4667 "supportedprofiles" Property of the /oic/sec/sp Resource.

4668 It indicates the OCF Device satisfies the normative security requirements for this document.

4669 When a device supports the baseline profile, the "supportedprofiles" Property shall contain sp-
4670 baseline-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.1.0", and may contain other
4671 profiles.

4672 When a manufacturer makes sp-baseline-v0 the default, by setting the "currentprofile" Property to
4673 "1.3.6.1.4.1.51414.0.0.1.0", the "supportedprofiles" Property shall contain sp-baseline-v0.

4674 **14.8.3.4 Security Profile Black (sp-black-v0)**

4675 **14.8.3.4.1 Black Profile General**

4676 The need for Security Profile Black v0 is to support devices and manufacturers who wish to
4677 certify their devices meeting this specific set of security criteria. A Device may satisfy the Black
4678 requirements as well as requirements of other profiles, the Black Security Profile is not
4679 necessarily mutually exclusive with other Security Profiles unless those requirements conflict with
4680 the explicit requirements of the Black Security Profile.

4681 **14.8.3.4.2 Devices Targeted for Security Profile Black v0**

4682 Security Profile Black devices could include any device a manufacturer wishes to certify at this
4683 profile, but healthcare devices and industrial devices with additional security requirements are
4684 the initial target. Additionally, manufacturers of devices at the edge of the network (or fog), or
4685 devices with exceptional profiles of trust bestowed upon them, may wish to certify at this profile;
4686 these types of devices may include, but are not limited to the following:

4687 – Bridges (Mapping devices between ecosystems handling virtual devices from different
4688 ecosystems)

4689 – Resource Directories (Devices trusted to manage OCF Security Domain resources)

4690 – Remote Access (Devices which have external access but can also act within the OCF
4691 Security Domain)

4692 – Healthcare Devices (Devices with specific requirements for enhanced security and privacy)

4693 – Industrial Devices (Devices with advanced management, security and attestation
4694 requirements)

4695 **14.8.3.4.3 Requirements for Certification at Security Profile Black (Normative)**

4696 Every device with "currentprofile" Property of the /oic/sec/sp Resource designating a Security
4697 Profile of "sp-black-v0", as defined in clause 14.8.2, must support each of the following:

- 4698 – Onboarding via OCF Rooted Certificate Chain, including PKI chain validation
- 4699 – Support for AES 128 encryption for data at rest and in transit.
- 4700 – Hardening minimums: manufacturer assertion of secure credential storage
- 4701 – In 7.1.2 in enumerated item #2: "The value should correspond to the value of
4702 subjectPublicKey defined in SubjectPublicKeyInfo of the certificate" is changed to require this
4703 format: "The value shall correspond to the value of subjectPublicKey defined in
4704 SubjectPublicKeyInfo of the certificate. "
- 4705 – In 7.1.2 in the enumerated item #3: "The value should correspond to the value of
4706 subjectPublicKey defined in SubjectPublicKeyInfo" is changed to require this format: "The
4707 value SHALL correspond to the value of subjectPublicKey defined in SubjectPublicKeyInfo."
- 4708 – In 14) in enumerated item #10 "The /oic/sec/cred Resource should contain credential(s) if
4709 required by the selected OTM" is changed to require the credential be stored: "The
4710 /oic/sec/cred Resource shall contain credential(s)."
- 4711 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4712 certificate and the extension's 'securityProfile' field shall contain sp-black-v0 represented by
4713 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.2.0".

4714 When a device supports the black profile, the "supportedprofiles" Property shall contain sp-black-
4715 v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.2.0", and may contain other profiles.

4716 When a manufacturer makes sp-black-v0 the default, by setting the "currentprofile" Property to
4717 "1.3.6.1.4.1.51414.0.0.2.0", the "supportedprofiles" Property shall contain sp-black-v0.

4718 The OCF Rooted Certificate Chain and PKI Is defined by and structured within a framework
4719 described in the supporting documents:

- 4720 – Certificate Profile (See 9.4.2)
- 4721 – Certificate Policy (see Certificate Policy document: OCF-TSC-SWG-CP-D03-171101.docx)

4722 **14.8.3.5 Security Profile Blue v0 (sp-blue-v0)**

4723 **14.8.3.5.1 Blue Profile General**

4724 The Security Profile Blue is used when manufacturers issue platform certificates for platforms
4725 containing manufacturer-embedded keys. Compatibility with interoperable trusted platforms is
4726 anticipated using certificate extensions defined by the Trusted Computing Group (TCG). OCF
4727 Security Domain owners evaluate manufacturer supplied certificates and attributed data to
4728 determine an appropriate OCF Security Profile that is configured for OCF Devices at onboarding.
4729 OCF Devices may satisfy multiple OCF Security Profiles. The OCF Security Domain owner may
4730 configure deployments using the Security Profile as OCF Security Domain partitioning criteria.

4731 Certificates issued to Blue Profile Devices shall be issued by a CA conforming to the CA Vetting
4732 Criteria defined by OCF.

4733 **14.8.3.5.2 Platforms and Devices for Security Profile Blue v0**

4734 The OCF Security Profile Blue anticipates an ecosystem where platform vendors may differ from
4735 OCF Device vendor and where platform vendors may implement trusted platforms that may
4736 conform to industry standards defining trusted platforms. The OCF Security Profile Blue specifies
4737 mechanisms for linking platforms with OCF Device(s) and for referencing quality assurance
4738 criteria produced by OCF conformance operations. The OCF Security Domain owner evaluates
4739 these data when an OCF Device is onboarded into the OCF Security Domain. Based on this

4740 evaluation the OCF Security Domain owner determines which Security Profile may be applied
4741 during OCF Device operation. All OCF Device types may be considered for evaluation using the
4742 OCF Security Profile Blue.

4743 **14.8.3.5.3 Requirements for Certification at Security Profile Blue v0**

4744 The OCF Device satisfies the Blue profile v0 (sp-blue-v0) when all of the security normative for
4745 this document version are satisfied and the following additional criteria are satisfied.

4746 OCF Blue profile defines the following OCF Device quality assurances:

- 4747 – The OCF Conformance criteria shall require vendor attestation that the conformant OCF
4748 Device was hosted on one or more platforms that satisfies OCF Blue platform security
4749 assurances and platform security and privacy functionality requirements.
- 4750 – The OCF Device achieving OCF Blue Security Profile compliance will be registered by OCF
4751 and published by OCF in a machine readable format.
- 4752 – The OCF Blue Security Profile compliance registry may be digitally signed by an OCF owned
4753 signing key.
- 4754 – The OCF Device shall include an X.509v3 OCF Compliance Extension (clause 9.4.2.2.4) in its
4755 certificate and the extension's 'securityProfile' field shall contain sp-blue-v0 represented by
4756 the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.3.0".
- 4757 – The OCF Device shall include an X.509v3 OCF CPL Attributes Extension (clause 9.4.2.2.7) in
4758 its certificate.
- 4759 – The OBT shall perform a lookup of the certification status of the OCF Device using the OCF
4760 CPL Attributes Extension values and verify that the sp-blue-v0 OID is listed in the extension's
4761 'securityprofiles' field.

4762 OCF Blue profile defines the following OCF Device security functionality:

- 4763 – OCF Device(s) shall be hosted on a platform where a cryptographic and secure storage
4764 functions are hardened by the platform.
- 4765 – OCF Device(s) hosted on a platform shall expose accompanying manufacturer credentials
4766 using the /oic/sec/cred Resource where the 'credusage' Property contains the value
4767 "oic.sec.cred.mfgcert".
- 4768 – OCF Device(s) that are hosted on a TCG-defined trusted platform should use an
4769 IEEE802.1AR IDDevID and should verify the "TCG Endorsement Key Credential". All TCG-
4770 defined manufacturer credentials may be identified by the "oic.sec.cred.mfgcert" value of the
4771 'credusage' Property of the /oic/sec/cred Resource. They may be used in response to
4772 selection of the "oic.sec.doxm.mfgcert" owner transfer method.
- 4773 – OCF Device(s) shall use AES128 equivalent minimum protection for transmitted data. (See
4774 NIST SP 800-57).
- 4775 – OCF Device(s) shall use AES128 equivalent minimum protection for stored data. (See NIST
4776 SP 800-57).
- 4777 – OCF Device(s) should use AES256 equivalent minimum protection for stored data. (See NIST
4778 SP 800-57).
- 4779 – OCF Device(s) should protect the /oic/sec/cred resource using the platform provided secure
4780 storage.
- 4781 – OCF Device(s) shall protect trust anchors (aka policy defining trusted CAs and pinned
4782 certificates) using platform provided secure storage.
- 4783 – OCF Device(s) should check certificate revocation status for locally issued certificates.

4784 – OCF onboarding tools (aka DOTS) shall check certificate revocation status for all certificates
4785 in manufacturer certificate path(s) if available. If a certificate is revoked, certificate validation
4786 fails and the connection is refused. The DOTS may disregard revocation status results if
4787 unavailable.

4788 OCF Blue profile defines the following platform security assurances:

4789 – Platforms implementing cryptographic service provider (CSP) functionality and secure storage
4790 functionality should be evaluated with a minimum FIPS140-2 Level 2 or Common Criteria EAL
4791 Level 2.

4792 – Platforms implementing trusted platform functionality should be evaluated with a minimum
4793 Common Criteria EAL Level 1.

4794 OCF Blue profile defines the following platform security and privacy functionality:

4795 – The Platform shall implement cryptographic service provider (CSP) functionality.

4796 – Platform CSP functionality shall include cryptographic algorithms, random number generation,
4797 secure time.

4798 – The Platform shall implement AES128 equivalent protection for transmitted data. (See NIST
4799 SP 800-57).

4800 – The Platform shall implement AES128 and AES256 equivalent protection for stored data. (See
4801 NIST SP 800-57).

4802 – Platforms hosting OCF Device(s) should implement a platform identifier following
4803 IEEE802.1AR or Trusted Computing Group(TCG) specifications.

4804 – Platforms based on Trusted Computing Group (TCG) platform definition that host OCF
4805 Device(s) should supply TCG-defined manufacture certificates; also known as "TCG
4806 Endorsement Key Credential" (which complies with IETF RFC 5280) and "TCG Platform
4807 Credential" (which complies with IETF RFC 5755).

4808 When a device supports the blue profile, the "supportedprofiles" Property shall contain sp-blue-v0,
4809 represented by the OID string "1.3.6.1.4.1.51414.0.0.3.0", and may contain other profiles.

4810 When a manufacturer makes sp-blue-v0 the default, by setting the "currentprofile" Property to
4811 "1.3.6.1.4.1.51414.0.0.3.0", the "supportedprofiles" Property shall contain sp-blue-v0.

4812 During onboarding, while the device state is RFOTM, the DOTS may update the "currentprofile"
4813 Property to one of the other values found in the "supportedprofiles" Property.

4814 **14.8.3.6 Security Profile Purple v0 (sp-purple-v0)**

4815 Every device with the /oic/sec/sp Resource designating "sp-purple-v0", as defined in clause
4816 14.8.2 must support following minimum requirements

4817 – Hardening minimums: secure credential storage, software integrity validation, secure update.

4818 – If a Certificate is used, the OCF Device shall include an X.509v3 OCF Compliance Extension
4819 (clause 9.4.2.2.4) in its certificate and the extension's 'securityProfile' field shall contain sp-
4820 purple-v0 represented by the ocfSecurityProfileOID string, "1.3.6.1.4.1.51414.0.0.4.0"

4821 – The OCF Device shall include a X.509v3 OCFPLAttributes Extension (clause 9.4.2.2.7) in its
4822 End-Entity Certificate when manufacturer certificate is used.

4823 Security Profile Purple has following optional security hardening requirements that the device can
4824 additionally support.

4825 – Hardening additions: secure boot, hardware backed secure storage

4826 – The OCF Device shall include a X.509v3 OCFSecurityClaims Extension (clause 9.4.2.2.6) in
4827 its End-Entity Certificate and it shall include corresponding OIDs to the hardening additions

4828 implemented and attested by the vendor. If there is no additional support for hardening
4829 requirements, X.509v3 OCFSecurityClaims Extension shall be omitted.

4830 For software integrity validation, OCF Device(s) shall provide the integrity validation mechanism
4831 for security critical executables such as cryptographic modules or secure service applications,
4832 and they should be validated before the execution. The key used for validating the integrity must
4833 be pinned at the least to the validating software module.

4834 For secure update, OCF Device(s) shall be able to update its firmware in a secure manner.

4835 For secure boot, OCF Device(s) shall implement the BIOS code (first-stage bootloader on ROM)
4836 to be executed by the processor on power-on, and secure boot parameters to be provisioned by
4837 tamper-proof memory. Also OCF Device(s) shall provide software module authentication for the
4838 security critical executables and stop the boot process if any integrity of them is compromised.

4839 For hardware backed secure storage, OCF Device(s) shall store sensitive data in non-volatile
4840 memory ("NVRAM") and prevent the retrieval of sensitive data through physical and/or electronic
4841 attacks.

4842 More details on security hardening guidelines for software integrity validation, secure boot,
4843 secure update, and hardware backed secure storage are described in 14.3, 14.5 and 14.2.2.2.

4844 Certificates issued to Purple Profile Devices shall be issued by a CA conforming to the CA
4845 Vetting Criteria defined by OCF.

4846 When a device supports the purple profile, the "supportedprofiles" Property shall contain sp-
4847 purple-v0, represented by the OID string "1.3.6.1.4.1.51414.0.0.4.0", and may contain other
4848 profiles.

4849 When a manufacturer makes sp-purple-v0 the default, by setting the "currentprofile" Property to
4850 "1.3.6.1.4.1.51414.0.0.4.0", the "supportedprofiles" Property shall contain sp-purple-v0.

4851 **15 Device Type Specific Requirements**

4852 **15.1 Bridging Security**

4853 **15.1.1 Universal Requirements for Bridging to another Ecosystem**

4854 The OCF Bridge Device shall go through OCF ownership transfer as any other onboarder would.

4855 The software of an OCF Bridge Device shall be field updatable. (This requirement need not be
4856 tested but can be certified via a vendor declaration.)

4857 Each Virtual OCF Device shall be onboarded by an OCF Onboarding tool. Each Virtual Bridged
4858 Device should be provisioned as appropriate in the Bridged Protocol. In other words, Virtual OCF
4859 Devices and Virtual Bridged Devices are treated the same way as physical Devices. They are
4860 entities that have to be provisioned in their network.

4861 Each Virtual OCF Device shall implement the behaviour required by ISO/IEC 30118-1:2018 and
4862 this document. Each Virtual OCF Device shall perform authentication, access control, and
4863 encryption according to the security settings it received from the Onboarding Tool. Each Virtual
4864 Bridged Device shall implement the security requirements of the Bridged Protocol.

4865 In addition, in order to be considered secure from an OCF perspective, the OCF Bridge Device
4866 implementation shall use appropriate ecosystem-specific security options for communication
4867 between the Virtual Bridged Devices instantiated by the OCF Bridge Device and Bridged Devices.
4868 This security shall include mutual authentication, and encryption and integrity protection of
4869 messages in the bridged ecosystem.

4870 A Virtual OCF Device may authenticate itself to the DOTS using the Manufacturer Certificate
4871 Based OTM (see clause 7.3.6) with the Manufacturer Certificate and corresponding private key of
4872 the OCF Bridge Device which instantiated that Virtual OCF Device.

4873 A Virtual OCF Device may authenticate itself to the OCF Cloud (see clause 10.5.2) using the
4874 Manufacturer Certificate and corresponding private key of the OCF Bridge Device which
4875 instantiated that Virtual OCF Device.

4876 **15.1.2 Additional Security Requirements specific to Bridged Protocols**

4877 **15.1.2.1 Additional Security Requirements specific to the AllJoyn Protocol**

4878 For AllJoyn translator, an OCF Onboarding Tool shall be able to block the communication of all
4879 OCF Devices with all Bridged Devices that don't communicate securely with the Bridge, by using
4880 the Bridge Device's "oic.r.securemode" Resource specified in ISO/IEC 30118-3:2018

4881 **15.1.2.2 Additional Security Requirements specific to the Bluetooth LE Protocol**

4882 The OCF Bridge Device shall implement oneM2M application access control as defined in the
4883 oneM2M Release 3 Specifications

4884 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices
4885 that don't communicate securely with the OCF Bridge Device

4886 **15.1.2.3 Additional Security Requirements specific to the U+ Protocol**

4887 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices
4888 that don't communicate securely with the OCF Bridge Device.

4889 **15.1.2.4 Additional Security Requirements specific to the Z-Wave Protocol**

4890 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices
4891 that don't communicate securely with the OCF Bridge Device.

4892 **15.1.2.5 Additional Security Requirements specific to the Zigbee Protocol**

4893 An OCF Bridge Device shall block the communication of all OCF Devices with all Bridged Devices
4894 that don't communicate securely with the OCF Bridge Device.

4895

4896

4897

4898

4899

4900

4901

4902

4903

4904

4905

4906

4907

4908

4909

4910

4911

4912

4913

4914

4915

4916 .

4917
4918
4919

Annex A (informative) Access Control Examples

4920

A.1 Example OCF ACL Resource

4921 Figure A-1 shows how a `/oic/sec/acl2` Resource could be configured to enforce an example
4922 access policy on the Server.

```
4923 {
4924   "aclist2": [
4925     {
4926       // Subject with ID ...01 should access two named Resources with access mode "CRUDN" (Create, Retrieve,
4927       Update, Delete and Notify)
4928       "subject": {"uuid": "XXX-...-XX01"},
4929       "resources": [
4930         {"href": "/oic/sh/light/1"},
4931         {"href": "/oic/sh/temp/0"}
4932       ],
4933       "permission": 31, // 31 dec = 0b0001 1111 which maps to ---N DURC
4934       "validity": [
4935         // The period starting at 18:00:00 UTC, on January 1, 2015 and
4936         // ending at 07:00:00 UTC on January 2, 2015
4937         "period": ["20150101T180000Z/20150102T070000Z"],
4938         // Repeats the {period} every week until the last day of Jan. 2015.
4939         "recurrence": ["RRULE:FREQ=WEEKLY;UNTIL=20150131T070000Z"]
4940       ],
4941       "aceid": 1
4942     }
4943   ],
4944   // An ACL provisioning and management service should be identified as
4945   // the resource owner
4946   "rowneruid": "0685B960-736F-46F7-BEC0-9E6CBD61ADC1"
4947 }
```

4948 **Figure A-1 – Example `/oic/sec/acl2` Resource**

4949

A.2 Example AMS

4950 Figure A-2 demonstrates how the `/oic/sec/amacl` Resource should be configured to achieve this
4951 objective.

```
4952 {
4953   "resources": [
4954     // If the {Subject} wants to access the /oic/sh/light/1 Resource at host1 and an Amacl was
4955     // supplied then use the sacl validation credential to enforce access.
4956     {"href": "/oic/sh/light/1"},
4957     // If the {Subject} wants to access the /oma/3 Resource at host2 and an AM sacl was
4958     // supplied then use the sacl validation credential to enforce access.
4959     {"href": "/oma/3"},
```

```
4960 // If the {Subject} wants to access any local Resource and an Amacl was supplied then use
4961 // the sacl validation credential to enforce access.
4962 {"wc": "**"}
4963 }
4964
```

Figure A-2 Example /oic/sec/amacl Resource

4965
4966
4967

Annex B (Informative) Execution Environment Security Profiles

4968 Given that IoT verticals and Devices will not be of uniform capabilities, a one-size-fits all security
4969 robustness requirements meeting all IOT applications and services will not serve the needs of
4970 OCF, and security profiles of varying degree of robustness (trustworthiness), cost and complexity
4971 have to be defined. To address a large ecosystem of vendors, the profiles can only be defined as
4972 requirements and the exact solutions meeting those requirements are specific to the vendors'
4973 open or proprietary implementations, and thus in most part outside scope of this document.

4974 To align with the rest of OCF documents, where Device classifications follow IETF RFC 7228
4975 (Terminology for constrained node networks) methodology, we limit the number of security
4976 profiles to a maximum of 3 (see Table B.1). However, our understanding is OCF capabilities
4977 criteria for each of 3 classes will be more fit to the current IoT chip market than that of IETF.

4978 Given the extremely low level of resources at class 0, our expectation is that class 0 Devices are
4979 either capable of no security functionality or easily breakable security that depend on
4980 environmental (e.g. availability of human) factors to perform security functions. This means the
4981 class 0 will not be equipped with an SEE.

4982

Table B.1 – OCF Security Profile

Platform class	SEE	Robustness level
0	No	N/A
1	Yes	Low
2	Yes	High

4983 NOTE This analysis acknowledges that these Platform classifications do not take into consideration of possibility of
4984 security co-processor or other hardware security capability that augments classification criteria (namely CPU speed,
4985 memory, storage).

4986
4987
4988

Annex C (normative) Resource Type definitions

4989 C.1 List of Resource Type definitions

4990 Table C.1 contains the list of defined security resources in this document.

4991 **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

4992

4993 C.2 Account Token

4994 C.2.1 Introduction

4995 Sign-up using generic account provider.

4996

4997 C.2.2 Well-known URI

4998 /oic/sec/account

4999 C.2.3 Resource type

5000 The resource type (rt) is defined as: ['oic.r.account'].

5001 C.2.4 OpenAPI 2.0 definition

```
5002 {  
5003   "swagger": "2.0",  
5004   "info": {  
5005     "title": "Account Token",  
5006     "version": "20190111",  
5007     "license": {  
5008       "name": "OCF Data Model License",  
5009       "url":  
5010 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
5011 CENSE.md",  
5012       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
5013 reserved."  
5014     }  
5015   }  
5016 }
```

```

5015     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5016   },
5017   "schemes": ["http"],
5018   "consumes": ["application/json"],
5019   "produces": ["application/json"],
5020   "paths": {
5021     "/oic/sec/account" : {
5022       "post": {
5023         "description": "Sign-up using generic account provider.\n",
5024         "parameters": [
5025           { "$ref": "#/parameters/interface" },
5026           {
5027             "name": "body",
5028             "in": "body",
5029             "required": true,
5030             "schema": { "$ref": "#/definitions/Account-request" },
5031             "x-example":
5032               {
5033                 "di" : "9cfbeb8e-5ale-4d1c-9d01-00c04fd430c8",
5034                 "authprovider" : "github",
5035                 "accesstoken" : "8802f2eaf8b5e147a936"
5036               }
5037           }
5038         ],
5039         "responses": {
5040           "204": {
5041             "description" : "2.04 Changed respond with required and optional information\n",
5042             "x-example":
5043               {
5044                 "rt": ["oic.r.account"],
5045                 "accesstoken" : "0f3d9f7fe5491d54077d",
5046                 "refreshtoken" : "00fe4644a6fbe5324eec",
5047                 "expiresin" : 3600,
5048                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
5049                 "redirecturi" : "coaps+tcp://example.com:443"
5050               },
5051             "schema": { "$ref": "#/definitions/Account-response" }
5052           }
5053         }
5054       },
5055       "delete": {
5056         "description": "Delete a device. This also removes all resources in the device on cloud
5057 side.\nexample: /oic/account?di=9cfbeb8e-5ale-4d1c-9d01-
5058 00c04fd430c8&accesstoken=0f3d9f7fe5491d54077d\n",
5059         "parameters": [
5060           { "$ref": "#/parameters/interface" }
5061         ],
5062         "responses": {
5063           "202": {
5064             "description" : "2.02 Deleted response informing the device is successfully
5065 deleted.\n"
5066           }
5067         }
5068       }
5069     },
5070     "parameters": {
5071       "interface" : {
5072         "in" : "query",
5073         "name" : "if",
5074         "type" : "string",
5075         "enum" : ["oic.if.baseline"]
5076       }
5077     },
5078     "definitions": {
5079       "Account-request" : {
5080         "properties": {
5081           "authprovider": {
5082             "description": "The name of Authorization Provider through which Access Token was
5083 obtained",
5084             "type": "string"

```

```

5086     },
5087     "accesstoken" : {
5088         "description": "Access-Token used for communication with OCF Cloud after account creation",
5089         "pattern": "(?!$|\\s+).*",
5090         "type": "string"
5091     },
5092     "di": {
5093         "description": "Format pattern according to IETF RFC 4122.",
5094         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5095         "type": "string"
5096     }
5097 },
5098 "type" : "object",
5099 "required": ["di", "accesstoken"]
5100 },
5101 "Account-response": {
5102     "properties": {
5103         "expiresin" : {
5104             "description": "Access-Token remaining life time in seconds (-1 if permanent)",
5105             "readOnly": true,
5106             "type": "integer"
5107         },
5108         "rt": {
5109             "description": "Resource Type of the Resource",
5110             "items": {
5111                 "maxLength": 64,
5112                 "type": "string",
5113                 "enum" : ["oic.r.account"]
5114             },
5115             "minItems": 1,
5116             "maxItems": 1,
5117             "readOnly": true,
5118             "type": "array"
5119         },
5120         "refreshtoken" : {
5121             "description": "Refresh token can be used to refresh the Access Token before getting
5122 expired",
5123             "pattern": "(?!$|\\s+).*",
5124             "readOnly": true,
5125             "type": "string"
5126         },
5127         "uid" : {
5128             "description": "Format pattern according to IETF RFC 4122.",
5129             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5130             "type": "string"
5131         },
5132         "accesstoken" : {
5133             "description": "Access-Token used for communication with cloud after account creation",
5134             "pattern": "(?!$|\\s+).*",
5135             "type": "string"
5136         },
5137         "n": {
5138             "$ref":
5139 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5140 schema.json#/definitions/n"
5141         },
5142         "id": {
5143             "$ref":
5144 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5145 schema.json#/definitions/id"
5146         },
5147         "redirecturi" : {
5148             "description": "Using this URI, the Client needs to reconnect to a redirected OCF Cloud.
5149 If provided, this value shall be used by the Device instead of Mediator-provided URI during the
5150 Device Registration.",
5151             "readOnly": true,
5152             "type": "string"
5153         },
5154         "if": {
5155             "description": "The interface set supported by this resource",
5156             "items": {

```

```

5157         "enum": [
5158             "oic.if.baseline"
5159         ],
5160         "type": "string"
5161     },
5162     "minItems": 1,
5163     "maxItems": 1,
5164     "uniqueItems": true,
5165     "readOnly": true,
5166     "type": "array"
5167 },
5168 },
5169 "type" : "object",
5170 "required": ["accesstoken", "refresh token", "expiresin", "uid"]
5171 }
5172 }
5173 }
5174

```

5175 C.2.5 Property definition

5176 Table C.2 defines the Properties that are part of the ['oic.r.account'] Resource Type

5177 **Table C.2 – The Property definitions of the Resource with type 'rt' = ['oic.r.account']**

Property name	Value type	Mandatory	Access mode	Description
refresh token	string	Yes	Read Only	Refresh token can be used to refresh the Access Token before getting expired
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
accesstoken	string	Yes	Read Write	Access-Token used for communication with cloud after account creation
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds (-1 if permanent)
rt	array: see schema	No	Read Only	Resource Type of the Resource
redirecturi	string	No	Read Only	Using this URI, the Client needs to reconnect to a redirected OCF Cloud. If provided, this value shall be used by the Device instead of Mediator-provided URI during the Device Registration.

uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
authprovider	string	No	Read Write	The name of Authorization Provider through which Access Token was obtained
di	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
accesstoken	string	Yes	Read Write	Access-Token used for communication with OCF Cloud after account creation

5178 **C.2.6 CRUDN behaviour**

5179 Table C.3 defines the CRUDN operations that are supported on the ['oic.r.account'] Resource
5180 Type

5181 **Table C.3 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.account']**

Create	Read	Update	Delete	Notify
		post	delete	

5182 **C.3 Access Control List**

5183 **C.3.1 Introduction**

5184 This Resource specifies the local access control list.
5185 When used without query parameters, all the ACE entries are returned.
5186 When used with a subjectuid, only the ACEs with the specified
5187 subjectuid are returned. If subjectuid and resources are specified,
5188 only the ACEs with the specified subjectuid and Resource hrefs are
5189 returned.
5190

5191 **C.3.2 Well-known URI**

5192 /oic/sec/acl

5193 **C.3.3 Resource type**

5194 The resource type (rt) is defined as: ['oic.r.acl'].

5195 **C.3.4 OpenAPI 2.0 definition**

```
5196 {
5197   "swagger": "2.0",
5198   "info": {
5199     "title": "Access Control List",
5200     "version": "v1.1-20161213",
5201     "license": {
5202       "name": "OCF Data Model License",
5203       "url":
```

```

5204 "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
5205 CENSE.md",
5206   "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
5207 reserved.",
5208   },
5209   "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
5210 },
5211 "schemes": ["http"],
5212 "consumes": ["application/json"],
5213 "produces": ["application/json"],
5214 "paths": {
5215   "/oic/sec/acl" : {
5216     "get": {
5217       "description": "This Resource specifies the local access control list.\nWhen used without
5218 query parameters, all the ACE entries are returned.\nWhen used with a subjectuuid, only the ACEs
5219 with the specified\nsubjectuuid are returned. If subjectuuid and resources are specified,\nonly the
5220 ACEs with the specified subjectuuid and Resource hrefs are\nreturned.\n",
5221       "parameters": [
5222         {"$ref": "#/parameters/interface"},
5223         {"$ref": "#/parameters/ace-filtered-uuid"},
5224         {"$ref": "#/parameters/ace-filtered-resources"}
5225       ],
5226       "responses": {
5227         "200": {
5228           "description": "",
5229           "x-example":
5230             {
5231               "rt": ["oic.r.acl"],
5232               "aclist": {
5233                 "aces": [
5234                   {
5235                     "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5236                     "resources": [
5237                       {
5238                         "href": "coaps://IP-ADDR/temp",
5239                         "rel": "some-rel",
5240                         "rt": ["oic.r.temperature"],
5241                         "if": ["oic.if.a"]
5242                       },
5243                       {
5244                         "href": "coaps://IP-ADDR/temp",
5245                         "rel": "some-rel",
5246                         "rt": ["oic.r.temperature"],
5247                         "if": ["oic.if.s"]
5248                       }
5249                     ],
5250                     "permission": 31,
5251                     "validity": [
5252                       {
5253                         "period": "20160101T180000Z/20170102T070000Z",
5254                         "recurrence": [ "DSTART:XXXXX",
5255 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5256                       },
5257                       {
5258                         "period": "20160101T180000Z/PT5H30M",
5259                         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5260                       }
5261                     ]
5262                   }
5263                 ]
5264               },
5265               "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5266             },
5267           "schema": { "$ref": "#/definitions/Acl" }
5268         },
5269         "400": {
5270           "description": "The request is invalid."
5271         }
5272       }
5273     },
5274     "post": {

```

```

5275     "description": "Updates the ACL resource with the provided values. ACEs provided\nin the
5276 update not currently in the ACL are added. ACEs that already\nexist in the ACL are ignored.\n\nNote
5277 that for the purposes of update, equivalency is determined\nby comparing the ACE subjectuid,
5278 permission, string comparisons\nof all validity elements, and string comparisons of all
5279 resource\nhrefs.\n",
5280     "parameters": [
5281         {"$ref": "#/parameters/interface"},
5282         {
5283             "name": "body",
5284             "in": "body",
5285             "required": true,
5286             "schema": { "$ref": "#/definitions/Acl" },
5287             "x-example":
5288                 {
5289                     "acllist": {
5290                         "aces": [
5291                             {
5292                                 "subjectuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
5293                                 "resources": [
5294                                     {
5295                                         "href": "coaps://IP-ADDR/temp",
5296                                         "rel": "some-rel",
5297                                         "rt": ["oic.r.temperature"],
5298                                         "if": ["oic.if.a"]
5299                                     },
5300                                     {
5301                                         "href": "coaps://IP-ADDR/temp",
5302                                         "rel": "some-rel",
5303                                         "rt": ["oic.r.temperature"],
5304                                         "if": ["oic.if.s"]
5305                                     }
5306                                 ],
5307                                 "permission": 31,
5308                                 "validity": [
5309                                     {
5310                                         "period": "20160101T180000Z/20170102T070000Z",
5311                                         "recurrence": [ "DSTART:XXXXX",
5312 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5313                                     },
5314                                     {
5315                                         "period": "20160101T180000Z/PT5H30M",
5316                                         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5317                                     }
5318                                 ]
5319                             }
5320                         ]
5321                     },
5322                     "rowneruid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5323                 }
5324             ]
5325         },
5326         "responses": {
5327             "400": {
5328                 "description": "The request is invalid."
5329             },
5330             "201": {
5331                 "description": "The ACL entry/entries is/are created."
5332             },
5333             "204": {
5334                 "description": "The ACL entry/entries is/are updated."
5335             }
5336         }
5337     },
5338     "delete": {
5339         "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5340 ACE entries are deleted.\nWhen DELETE is used with a subjectuid, only the ACEs with the
5341 specified\nsubjectuid are deleted. If subjectuid and resources are specified,\nonly the ACEs with
5342 the specified subjectuid and resource hrefs are\ndeleted.\n",
5343         "parameters": [
5344             {"$ref": "#/parameters/interface"},
5345             {"$ref": "#/parameters/ace-filtered-uuid"},

```

```

5346         {"$ref": "#/parameters/ace-filtered-resources"}
5347     ],
5348     "responses": {
5349         "200": {
5350             "description": "The matching ACEs or the entire ACL resource has been successfully
5351 deleted."
5352         },
5353         "400": {
5354             "description": "The request is invalid."
5355         }
5356     }
5357 },
5358 },
5359 },
5360 "parameters": {
5361     "interface": {
5362         "in": "query",
5363         "name": "if",
5364         "type": "string",
5365         "enum": ["oic.if.baseline"]
5366     },
5367     "ace-filtered-uuid": {
5368         "in": "query",
5369         "name": "subjectuuid",
5370         "required": false,
5371         "type": "string",
5372         "description": "Only applies to ACEs with the specified subject UUID",
5373         "x-example": "se61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5374     },
5375     "ace-filtered-resources": {
5376         "in": "query",
5377         "name": "resources",
5378         "required": false,
5379         "type": "string",
5380         "description": "Only applies to ACEs with the specified subhjectuuid | and resources href",
5381         "x-example": "coaps://IP-ADDR/temp"
5382     }
5383 },
5384 "definitions": {
5385     "Acl": {
5386         "properties": {
5387             "owneruuid": {
5388                 "description": "The value identifies the unique resource owner\nFormat pattern according
5389 to IETF RFC 4122.",
5390                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
5391                 "type": "string"
5392             },
5393             "rt": {
5394                 "description": "Resource Type of the Resource",
5395                 "items": {
5396                     "maxLength": 64,
5397                     "type": "string",
5398                     "enum": ["oic.r.acl"]
5399                 },
5400                 "minItems": 1,
5401                 "readOnly": true,
5402                 "type": "array"
5403             },
5404             "aclist": {
5405                 "description": "Subject-based Access Control Entries in the ACL resource",
5406                 "properties": {
5407                     "aces": {
5408                         "items": {
5409                             "properties": {
5410                                 "permission": {
5411                                     "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask
5412 indicating permissions",
5413                                     "x-detail-desc": [
5414                                         "0 - No permissions",
5415                                         "1 - Create permission is granted",
5416                                         "2 - Read, observe, discover permission is granted",

```

```

5417         "4 - Write, update permission is granted",
5418         "8 - Delete permission is granted",
5419         "16 - Notify permission is granted"
5420     ],
5421     "maximum": 31,
5422     "minimum": 0,
5423     "type": "integer"
5424 },
5425 "resources": {
5426     "description": "References the application's resources to which a security
policy applies",
5427     "items": {
5428         "properties": {
5429             "anchor": {
5430                 "description": "This is used to override the context URI e.g. override the
5431 URI of the containing collection.",
5432                 "format": "uri",
5433                 "maxLength": 256,
5434                 "type": "string"
5435             },
5436             "di": {
5437                 "description": "The device ID\nFormat pattern according to IETF RFC 4122.",
5438                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-
5439 [a-fA-F0-9]{12}$",
5440                 "type": "string"
5441             },
5442             "eps": {
5443                 "description": "the Endpoint information of the target Resource",
5444                 "items": {
5445                     "properties": {
5446                         "ep": {
5447                             "description": "Transport Protocol Suite + Endpoint Locator",
5448                             "format": "uri",
5449                             "type": "string"
5450                         },
5451                         "pri": {
5452                             "description": "The priority among multiple Endpoints",
5453                             "minimum": 1,
5454                             "type": "integer"
5455                         }
5456                     }
5457                 },
5458                 "type": "object"
5459             },
5460             "type": "array"
5461         },
5462         "href": {
5463             "description": "This is the target URI, it can be specified as a Relative
5464 Reference or fully-qualified URI.",
5465             "format": "uri",
5466             "maxLength": 256,
5467             "type": "string"
5468         },
5469         "if": {
5470             "description": "The interface set supported by this resource",
5471             "items": {
5472                 "enum": [
5473                     "oic.if.baseline",
5474                     "oic.if.ll",
5475                     "oic.if.b",
5476                     "oic.if.rw",
5477                     "oic.if.r",
5478                     "oic.if.a",
5479                     "oic.if.s"
5480                 ],
5481                 "type": "string"
5482             },
5483             "minItems": 1,
5484             "type": "array"
5485         },
5486         "ins": {
5487             "description": "The instance identifier for this web link in an array of

```

```

5488 web links - used in collections",
5489     "type": "integer"
5490   },
5491   "p": {
5492     "description": "Specifies the framework policies on the Resource
5493 referenced by the target URI",
5494     "properties": {
5495       "bm": {
5496         "description": "Specifies the framework policies on the Resource
5497 referenced by the target URI for e.g. observable and discoverable",
5498         "type": "integer"
5499       }
5500     },
5501     "required": [
5502       "bm"
5503     ],
5504     "type": "object"
5505   },
5506   "rel": {
5507     "description": "The relation of the target URI referenced by the link to
5508 the context URI",
5509     "oneOf": [
5510       {
5511         "default": [
5512           "hosts"
5513         ],
5514         "items": {
5515           "maxLength": 64,
5516           "type": "string"
5517         },
5518         "minItems": 1,
5519         "type": "array"
5520       },
5521       {
5522         "default": "hosts",
5523         "maxLength": 64,
5524         "type": "string"
5525       }
5526     ]
5527   },
5528   "rt": {
5529     "description": "Resource Type of the Resource",
5530     "items": {
5531       "maxLength": 64,
5532       "type": "string"
5533     },
5534     "minItems": 1,
5535     "type": "array"
5536   },
5537   "title": {
5538     "description": "A title for the link relation. Can be used by the UI to
5539 provide a context.",
5540     "maxLength": 64,
5541     "type": "string"
5542   },
5543   "type": {
5544     "default": "application/cbor",
5545     "description": "A hint at the representation of the resource referenced by
5546 the target URI. This represents the media types that are used for both accepting and emitting.",
5547     "items": {
5548       "maxLength": 64,
5549       "type": "string"
5550     },
5551     "minItems": 1,
5552     "type": "array"
5553   }
5554 },
5555 "required": [
5556   "href",
5557   "rt",
5558   "if"

```

```

5559         ],
5560         "type": "object"
5561     },
5562     "type": "array"
5563 },
5564 "subjectuuid": {
5565     "anyOf": [
5566         {
5567             "description": "The id of the device to which the ace applies to or \"*\
5568 for anonymous access",
5569             "pattern": "^\\*$",
5570             "type": "string"
5571         },
5572         {
5573             "description": "Format pattern according to IETF RFC 4122.",
5574             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5575 fA-F0-9]{12}$",
5576             "type": "string"
5577         }
5578     ]
5579 },
5580 "validity": {
5581     "description": "validity is an array of time-pattern objects",
5582     "items": {
5583         "description": "The time-pattern contains a period and recurrence expressed in
5584 IETF RFC 5545 syntax",
5585         "properties": {
5586             "period": {
5587                 "description": "String represents a period using the IETF RFC 5545 Period",
5588                 "type": "string"
5589             },
5590             "recurrence": {
5591                 "description": "String array represents a recurrence rule using the IETF
5592 RFC 5545 Recurrence",
5593                 "items": {
5594                     "type": "string"
5595                 },
5596                 "type": "array"
5597             }
5598         },
5599         "required": [
5600             "period"
5601         ],
5602         "type": "object"
5603     },
5604     "type": "array"
5605 }
5606 },
5607 "required": [
5608     "resources",
5609     "permission",
5610     "subjectuuid"
5611 ],
5612 "type": "object"
5613 },
5614 "type": "array"
5615 }
5616 },
5617 "required": [
5618     "aces"
5619 ],
5620 "type": "object"
5621 },
5622 "n": {
5623     "$ref":
5624 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
5625 schema.json#/definitions/n"
5626 },
5627 "id": {
5628     "$ref":
5629 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-

```

```

5630 schema.json#/definitions/id"
5631     },
5632     "if": {
5633         "description": "The interface set supported by this resource",
5634         "items": {
5635             "enum": [
5636                 "oic.if.baseline"
5637             ],
5638             "type": "string"
5639         },
5640         "minItems": 1,
5641         "readOnly": true,
5642         "type": "array"
5643     }
5644 },
5645 "type" : "object",
5646 "required": ["aclist", "rowneruuid"]
5647 }
5648 }
5649 }
5650

```

5651 C.3.5 Property definition

5652 Table C.4 defines the Properties that are part of the ['oic.r.acl'] Resource Type

5653 **Table C.4 – The Property definitions of the Resource with type 'rt' = ['oic.r.acl']**

Property name	Value type	Mandatory	Access mode	Description
rowneruuid	string	Yes	Read Write	The value identifies the unique resource owner Format pattern according to IETF RFC 4122.
aclist	object: see schema	Yes	Read Write	Subject-based Access Control Entries in the ACL resource
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	

5654 C.3.6 CRUDN behaviour

5655 Table C.5 defines the CRUDN operations that are supported on the ['oic.r.acl'] Resource Type

5656 **Table C.5 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

5657 C.4 Access Control List-2

5658 C.4.1 Introduction

5659 This Resource specifies the local access control list.
5660 When used without query parameters, all the ACE entries are returned.
5661 When used with a query parameter, only the ACEs matching the specified
5662 parameter are returned.
5663

5664 C.4.2 Well-known URI

5665 /oic/sec/acl2

5666 C.4.3 Resource type

5667 The resource type (rt) is defined as: ['oic.r.acl2'].

5668 C.4.4 OpenAPI 2.0 definition

```
5669 {  
5670   "swagger": "2.0",  
5671   "info": {  
5672     "title": "Access Control List-2",  
5673     "version": "20190111",  
5674     "license": {  
5675       "name": "OCF Data Model License",  
5676       "url":  
5677         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI  
5678         CENSE.md",  
5679       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights  
5680       reserved."  
5681     },  
5682     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"  
5683   },  
5684   "schemes": ["http"],  
5685   "consumes": ["application/json"],  
5686   "produces": ["application/json"],  
5687   "paths": {  
5688     "/oic/sec/acl2" : {  
5689       "get": {  
5690         "description": "This Resource specifies the local access control list.\nWhen used without  
5691         query parameters, all the ACE entries are returned.\nWhen used with a query parameter, only the ACEs  
5692         matching the specified\nparameter are returned.\n",  
5693         "parameters": [  
5694           {"$ref": "#/parameters/interface"},  
5695           {"$ref": "#/parameters/ace-filtered"}  
5696         ],  
5697         "responses": {  
5698           "200": {  
5699             "description": "",  
5700             "x-example":  
5701               {  
5702                 "rt" : ["oic.r.acl2"],  
5703                 "aclist2": [  
5704                   {  
5705                     "aceid": 1,  
5706                     "subject": {  
5707                       "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",  
5708                       "role": "SOME_STRING"  
5709                     },  
5710                     "resources": [  
5711                       {  
5712                         "href": "/light",  
5713                         "rt": ["oic.r.light"],  
5714                         "if": ["oic.if.baseline", "oic.if.a"]  
5715                       },  
5716                       {  
5717                         "href": "/door",  
5718                         "rt": ["oic.r.door"],  
5719                         "if": ["oic.if.baseline", "oic.if.a"]
```

```

5720     }
5721   ],
5722   "permission": 24
5723 },
5724 {
5725   "aceid": 2,
5726   "subject": {
5727     "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5728   },
5729   "resources": [
5730     {
5731       "href": "/light",
5732       "rt": ["oic.r.light"],
5733       "if": ["oic.if.baseline", "oic.if.a"]
5734     },
5735     {
5736       "href": "/door",
5737       "rt": ["oic.r.door"],
5738       "if": ["oic.if.baseline", "oic.if.a"]
5739     }
5740   ],
5741   "permission": 24
5742 },
5743 {
5744   "aceid": 3,
5745   "subject": {"conntype": "anon-clear"},
5746   "resources": [
5747     {
5748       "href": "/light",
5749       "rt": ["oic.r.light"],
5750       "if": ["oic.if.baseline", "oic.if.a"]
5751     },
5752     {
5753       "href": "/door",
5754       "rt": ["oic.r.door"],
5755       "if": ["oic.if.baseline", "oic.if.a"]
5756     }
5757   ],
5758   "permission": 16,
5759   "validity": [
5760     {
5761       "period": "20160101T180000Z/20170102T070000Z",
5762       "recurrence": [ "DSTART:XXXXX",
5763 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5764     },
5765     {
5766       "period": "20160101T180000Z/PT5H30M",
5767       "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
5768     }
5769   ]
5770 },
5771 ],
5772 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
5773 },
5774 "schema": { "$ref": "#/definitions/Acl2" }
5775 },
5776 "400": {
5777   "description": "The request is invalid."
5778 }
5779 },
5780 },
5781 "post": {
5782   "description": "Updates the ACL resource with the provided ACEs.\n\nACEs provided in the
5783 update with aceids not currently in the ACL\nresource are added.\n\nACEs provided in the update with
5784 aceid(s) already in the ACL completely\nreplace the ACE(s) in the ACL resource.\n\nACEs provided in
5785 the update without aceid properties are added and\nassigned unique aceids in the ACL resource.\n",
5786   "parameters": [
5787     { "$ref": "#/parameters/interface" },
5788     { "$ref": "#/parameters/ace-filtered" },
5789     {
5790       "name": "body",

```

```

5791     "in": "body",
5792     "required": true,
5793     "schema": { "$ref": "#/definitions/Acl2-Update" },
5794     "x-example":
5795     {
5796         "acllist2": [
5797             {
5798                 "aceid": 1,
5799                 "subject": {
5800                     "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
5801                     "role": "SOME_STRING"
5802                 },
5803                 "resources": [
5804                     {
5805                         "href": "/light",
5806                         "rt": ["oic.r.light"],
5807                         "if": ["oic.if.baseline", "oic.if.a"]
5808                     },
5809                     {
5810                         "href": "/door",
5811                         "rt": ["oic.r.door"],
5812                         "if": ["oic.if.baseline", "oic.if.a"]
5813                     }
5814                 ],
5815                 "permission": 24
5816             },
5817             {
5818                 "aceid": 3,
5819                 "subject": {
5820                     "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5821                 },
5822                 "resources": [
5823                     {
5824                         "href": "/light",
5825                         "rt": ["oic.r.light"],
5826                         "if": ["oic.if.baseline", "oic.if.a"]
5827                     },
5828                     {
5829                         "href": "/door",
5830                         "rt": ["oic.r.door"],
5831                         "if": ["oic.if.baseline", "oic.if.a"]
5832                     }
5833                 ],
5834                 "permission": 24
5835             }
5836         ],
5837         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
5838     }
5839 }
5840 ],
5841 "responses": {
5842     "400": {
5843         "description": "The request is invalid."
5844     },
5845     "201": {
5846         "description": "The ACL entry is created."
5847     },
5848     "204": {
5849         "description": "The ACL entry is updated."
5850     }
5851 }
5852 },
5853 "delete": {
5854     "description": "Deletes ACL entries.\nWhen DELETE is used without query parameters, all the
5855 ACE entries are deleted.\nWhen DELETE is used with a query parameter, only the ACEs matching
5856 the\nspecified parameter are deleted.\n",
5857     "parameters": [
5858         {"$ref": "#/parameters/interface"},
5859         {"$ref": "#/parameters/ace-filtered"}
5860     ],
5861     "responses": {

```

```

5862         "200": {
5863             "description" : "The matching ACEs or the entire ACL resource has been successfully
5864 deleted."
5865         },
5866         "400": {
5867             "description" : "The request is invalid."
5868         }
5869     }
5870 }
5871 }
5872 },
5873 "parameters": {
5874     "interface" : {
5875         "in" : "query",
5876         "name" : "if",
5877         "type" : "string",
5878         "enum" : ["oic.if.baseline"]
5879     },
5880     "ace-filtered" : {
5881         "in" : "query",
5882         "name" : "aceid",
5883         "required" : false,
5884         "type" : "integer",
5885         "description" : "Only applies to the ACE with the specified aceid",
5886         "x-example" : 2112
5887     }
5888 },
5889 "definitions": {
5890     "Acl2" : {
5891         "properties": {
5892             "rowneruuid" : {
5893                 "description": "The value identifies the unique resource owner\nFormat pattern according
5894 to IETF RFC 4122.",
5895                 "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}$",
5896                 "type": "string"
5897             },
5898             "rt" : {
5899                 "description": "Resource Type of the Resource",
5900                 "items": {
5901                     "maxLength": 64,
5902                     "type": "string",
5903                     "enum": ["oic.r.acl2"]
5904                 },
5905                 "minItems": 1,
5906                 "maxItems": 1,
5907                 "readOnly": true,
5908                 "type": "array"
5909             },
5910             "aclist2" : {
5911                 "description": "Access Control Entries in the ACL resource",
5912                 "items": {
5913                     "properties": {
5914                         "aceid": {
5915                             "description": "An identifier for the ACE that is unique within the ACL. In cases
5916 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
5917                             "minimum": 1,
5918                             "type": "integer"
5919                         },
5920                         "permission": {
5921                             "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
5922 permissions",
5923                             "x-detail-desc": [
5924                                 "0 - No permissions",
5925                                 "1 - Create permission is granted",
5926                                 "2 - Read, observe, discover permission is granted",
5927                                 "4 - Write, update permission is granted",
5928                                 "8 - Delete permission is granted",
5929                                 "16 - Notify permission is granted"
5930                             ],
5931                             "maximum": 31,
5932                             "minimum": 0,

```

```

5933         "type": "integer"
5934     },
5935     "resources": {
5936         "description": "References the application's resources to which a security policy
5937 applies",
5938         "items": {
5939             "description": "Each resource must have at least one of these properties set",
5940             "properties": {
5941                 "href": {
5942                     "description": "When present, the ACE only applies when the href matches\nThis
5943 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
5944                     "format": "uri",
5945                     "maxLength": 256,
5946                     "type": "string"
5947                 },
5948                 "if": {
5949                     "description": "When present, the ACE only applies when the if (interface)
5950 matches\nThe interface set supported by this resource",
5951                     "items": {
5952                         "enum": [
5953                             "oic.if.baseline",
5954                             "oic.if.ll",
5955                             "oic.if.b",
5956                             "oic.if.rw",
5957                             "oic.if.r",
5958                             "oic.if.a",
5959                             "oic.if.s"
5960                         ],
5961                         "type": "string"
5962                     },
5963                     "minItems": 1,
5964                     "type": "array"
5965                 },
5966                 "rt": {
5967                     "description": "When present, the ACE only applies when the rt (resource type)
5968 matches\nResource Type of the Resource",
5969                     "items": {
5970                         "maxLength": 64,
5971                         "type": "string"
5972                     },
5973                     "minItems": 1,
5974                     "type": "array"
5975                 },
5976                 "wc": {
5977                     "description": "A wildcard matching policy",
5978                     "pattern": "^[+*]$",
5979                     "type": "string"
5980                 }
5981             },
5982             "type": "object"
5983         },
5984         "type": "array"
5985     },
5986     "subject": {
5987         "anyOf": [
5988             {
5989                 "description": "Device identifier",
5990                 "properties": {
5991                     "uuid": {
5992                         "description": "A UUID Device ID\nFormat pattern according to IETF RFC
5993 4122.",
5994                         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
5995 fA-F0-9]{12}$",
5996                         "type": "string"
5997                     }
5998                 },
5999                 "required": [
6000                     "uuid"
6001                 ],
6002                 "type": "object"
6003             },

```

```

6004         {
6005             "description": "Security role specified as an <Authority> & <Rolename>. A NULL
6006 <Authority> refers to the local entity or device.",
6007             "properties": {
6008                 "authority": {
6009                     "description": "The Authority component of the entity being identified. A
6010 NULL <Authority> refers to the local entity or device.",
6011                     "type": "string"
6012                 },
6013                 "role": {
6014                     "description": "The ID of the role being identified.",
6015                     "type": "string"
6016                 }
6017             },
6018             "required": [
6019                 "role"
6020             ],
6021             "type": "object"
6022         },
6023         {
6024             "properties": {
6025                 "conntype": {
6026                     "description": "This property allows an ACE to be matched based on the
6027 connection or message type",
6028                     "x-detail-desc": [
6029                         "auth-crypt - ACE applies if the Client is authenticated and the data
6030 channel or message is encrypted and integrity protected",
6031                         "anon-clear - ACE applies if the Client is not authenticated and the data
6032 channel or message is not encrypted but may be integrity protected"
6033                     ],
6034                     "enum": [
6035                         "auth-crypt",
6036                         "anon-clear"
6037                     ],
6038                     "type": "string"
6039                 }
6040             },
6041             "required": [
6042                 "conntype"
6043             ],
6044             "type": "object"
6045         }
6046     ],
6047 },
6048 "validity": {
6049     "description": "validity is an array of time-pattern objects",
6050     "items": {
6051         "description": "The time-pattern contains a period and recurrence expressed in
6052 IETF RFC 5545 syntax",
6053         "properties": {
6054             "period": {
6055                 "description": "String represents a period using the RFC5545 Period",
6056                 "type": "string"
6057             },
6058             "recurrence": {
6059                 "description": "String array represents a recurrence rule using the IETF RFC
6060 5545 Recurrence",
6061                 "items": {
6062                     "type": "string"
6063                 },
6064                 "type": "array"
6065             }
6066         },
6067         "required": [
6068             "period"
6069         ],
6070         "type": "object"
6071     },
6072     "type": "array"
6073 },
6074 },

```

```

6075         "required": [
6076             "aceid",
6077             "resources",
6078             "permission",
6079             "subject"
6080         ],
6081         "type": "object"
6082     },
6083     "type": "array"
6084 },
6085 "n": {
6086     "$ref":
6087 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6088 schema.json#/definitions/n"
6089 },
6090 "id": {
6091     "$ref":
6092 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6093 schema.json#/definitions/id"
6094 },
6095 "if" : {
6096     "description": "The interface set supported by this resource",
6097     "items": {
6098         "enum": [
6099             "oic.if.baseline"
6100         ],
6101         "type": "string"
6102     },
6103     "minItems": 1,
6104     "maxItems": 1,
6105     "readOnly": true,
6106     "type": "array"
6107 }
6108 },
6109 "type" : "object",
6110 "required": ["aclist2", "rowneruuid"]
6111 },
6112 "Acl2-Update" : {
6113     "properties": {
6114         "rowneruuid" : {
6115             "description": "The value identifies the unique resource owner\n Format pattern according
6116 to IETF RFC 4122.",
6117             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
6118 9]{12}$",
6119             "type": "string"
6120         },
6121         "aclist2" : {
6122             "description": "Access Control Entries in the ACL resource",
6123             "items": {
6124                 "properties": {
6125                     "aceid": {
6126 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
6127                     "minimum": 1,
6128                     "type": "integer"
6129                 },
6130             },
6131             "permission": {
6132                 "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
6133 permissions",
6134                 "x-detail-desc": [
6135                     "0 - No permissions",
6136                     "1 - Create permission is granted",
6137                     "2 - Read, observe, discover permission is granted",
6138                     "4 - Write, update permission is granted",
6139                     "8 - Delete permission is granted",
6140                     "16 - Notify permission is granted"
6141                 ],
6142                 "maximum": 31,
6143                 "minimum": 0,
6144                 "type": "integer"
6145             },

```

```

6146         "resources": {
6147             "description": "References the application's resources to which a security policy
6148 applies",
6149             "items": {
6150                 "description": "Each resource must have at least one of these properties set",
6151                 "properties": {
6152                     "href": {
6153                         "description": "When present, the ACE only applies when the href matches\nThis
6154 is the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6155                         "format": "uri",
6156                         "maxLength": 256,
6157                         "type": "string"
6158                     },
6159                     "if": {
6160                         "description": "When present, the ACE only applies when the if (interface)
6161 matches\nThe interface set supported by this resource",
6162                         "items": {
6163                             "enum": [
6164                                 "oic.if.baseline",
6165                                 "oic.if.ll",
6166                                 "oic.if.b",
6167                                 "oic.if.rw",
6168                                 "oic.if.r",
6169                                 "oic.if.a",
6170                                 "oic.if.s"
6171                             ],
6172                             "type": "string"
6173                         },
6174                         "minItems": 1,
6175                         "type": "array"
6176                     },
6177                     "rt": {
6178                         "description": "When present, the ACE only applies when the rt (resource type)
6179 matches\nResource Type of the Resource",
6180                         "items": {
6181                             "maxLength": 64,
6182                             "type": "string"
6183                         },
6184                         "minItems": 1,
6185                         "type": "array"
6186                     },
6187                     "wc": {
6188                         "description": "A wildcard matching policy",
6189                         "x-detail-desc": [
6190                             "+ - Matches all discoverable resources",
6191                             "- - Matches all non-discoverable resources",
6192                             "* - Matches all resources"
6193                         ],
6194                         "enum": [
6195                             "+",
6196                             "-",
6197                             "*"
6198                         ],
6199                         "type": "string"
6200                     }
6201                 },
6202                 "type": "object"
6203             },
6204             "type": "array"
6205         },
6206         "subject": {
6207             "anyOf": [
6208                 {
6209                     "description": "Device identifier",
6210                     "properties": {
6211                         "uuid": {
6212                             "description": "A UUID Device ID\nFormat pattern according to IETF RFC
6213 4122.",
6214                             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
6215 fA-F0-9]{12}$",
6216                             "type": "string"

```

```

6217         }
6218     },
6219     "required": [
6220         "uuid"
6221     ],
6222     "type": "object"
6223 },
6224 {
6225     "description": "Security role specified as an <Authority> & <Rolename>. A NULL
6226 <Authority> refers to the local entity or device.",
6227     "properties": {
6228         "authority": {
6229             "description": "The Authority component of the entity being identified. A
6230 NULL <Authority> refers to the local entity or device.",
6231             "type": "string"
6232         },
6233         "role": {
6234             "description": "The ID of the role being identified.",
6235             "type": "string"
6236         }
6237     },
6238     "required": [
6239         "role"
6240     ],
6241     "type": "object"
6242 },
6243 {
6244     "properties": {
6245         "conntype": {
6246             "description": "This property allows an ACE to be matched based on the
6247 connection or message type",
6248             "x-detail-desc": [
6249                 "auth-crypt - ACE applies if the Client is authenticated and the data
6250 channel or message is encrypted and integrity protected",
6251                 "anon-clear - ACE applies if the Client is not authenticated and the data
6252 channel or message is not encrypted but may be integrity protected"
6253             ],
6254             "enum": [
6255                 "auth-crypt",
6256                 "anon-clear"
6257             ],
6258             "type": "string"
6259         }
6260     },
6261     "required": [
6262         "conntype"
6263     ],
6264     "type": "object"
6265 }
6266 ]
6267 },
6268 "validity": {
6269     "description": "validity is an array of time-pattern objects",
6270     "items": {
6271         "description": "The time-pattern contains a period and recurrence expressed in
6272 IETF RFC 5545 syntax",
6273         "properties": {
6274             "period": {
6275                 "description": "String represents a period using the RFC5545 Period",
6276                 "type": "string"
6277             },
6278             "recurrence": {
6279                 "description": "String array represents a recurrence rule using the IETF RFC
6280 5545 Recurrence",
6281                 "items": {
6282                     "type": "string"
6283                 },
6284                 "type": "array"
6285             }
6286         },
6287     "required": [

```

```

6288         "period"
6289     ],
6290     "type": "object"
6291 },
6292 "type": "array"
6293 }
6294 },
6295 "required": [
6296     "resources",
6297     "permission",
6298     "subject"
6299 ],
6300 "type": "object"
6301 },
6302 "type": "array"
6303 }
6304 },
6305 "type" : "object"
6306 }
6307 }
6308 }
6309

```

6310 C.4.5 Property definition

6311 Table C.6 defines the Properties that are part of the ['oic.r.acl2'] Resource Type

6312 **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	Yes	Read Write	Access Control Entries in the ACL resource
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruuid	string	Yes	Read Write	The value identifies the unique resource owner Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
aclist2	array: see schema	No	Read Write	Access Control Entries in the ACL resource
rowneruuid	string	No	Read Write	The value identifies the unique resource owner Format pattern according to IETF RFC 4122.

6313 **C.4.6 CRUDN behaviour**

6314 Table C.7 defines the CRUDN operations that are supported on the ['oic.r.acl2'] Resource Type

6315 **Table C.7 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.acl2']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

6316 **C.5 Managed Access Control**

6317 **C.5.1 Introduction**

6318 This resource specifies the host resources with access permission that is managed by an AMS.

6319

6320 **C.5.2 Well-known URI**

6321 /oic/sec/amacl

6322 **C.5.3 Resource type**

6323 The resource type (rt) is defined as: ['oic.r.amacl'].

6324 **C.5.4 OpenAPI 2.0 definition**

```

6325 {
6326   "swagger": "2.0",
6327   "info": {
6328     "title": "Managed Access Control",
6329     "version": "20190111",
6330     "license": {
6331       "name": "OCF Data Model License",
6332       "url":
6333         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6334         CENSE.md",
6335       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6336         reserved."
6337     },
6338     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6339   },
6340   "schemes": ["http"],
6341   "consumes": ["application/json"],
6342   "produces": ["application/json"],
6343   "paths": {
6344     "/oic/sec/amacl" : {
6345       "get": {
6346         "description": "This resource specifies the host resources with access permission that is
6347         managed by an AMS.\n",
6348         "parameters": [
6349           {"$ref": "#/parameters/interface"}
6350         ],
6351         "responses": {
6352           "200": {
6353             "description": "",
6354             "x-example":
6355               {
6356                 "rt" : ["oic.r.amacl"],
6357                 "resources": [
6358                   {
6359                     "href": "/temp",
6360                     "rt": ["oic.r.temperature"],
6361                     "if": ["oic.if.baseline", "oic.if.a"]
6362                   },
6363                   {
6364                     "href": "/temp",
6365                     "rt": ["oic.r.temperature"],
6366                     "if": ["oic.if.baseline", "oic.if.s"]
6367                   }
6368                 ]
6369               }
6370         }
6371       }
6372     }
6373   }

```

```

6370         "schema": { "$ref": "#/definitions/Amacl" }
6371     }
6372 },
6373 },
6374 "post": {
6375     "description": "Sets the new amacl data\n",
6376     "parameters": [
6377         {"$ref": "#/parameters/interface"},
6378     ],
6379     "name": "body",
6380     "in": "body",
6381     "required": true,
6382     "schema": { "$ref": "#/definitions/Amacl" },
6383     "x-example":
6384     {
6385         "resources": [
6386             {
6387                 "href": "/temp",
6388                 "rt": ["oic.r.temperature"],
6389                 "if": ["oic.if.baseline", "oic.if.a"]
6390             },
6391             {
6392                 "href": "/temp",
6393                 "rt": ["oic.r.temperature"],
6394                 "if": ["oic.if.baseline", "oic.if.s"]
6395             }
6396         ]
6397     }
6398 },
6399 ],
6400 "responses": {
6401     "400": {
6402         "description": "The request is invalid."
6403     },
6404     "201": {
6405         "description": "The AMACL entry is created."
6406     },
6407     "204": {
6408         "description": "The AMACL entry is updated."
6409     }
6410 },
6411 },
6412 "put": {
6413     "description": "Creates the new acl data\n",
6414     "parameters": [
6415         {"$ref": "#/parameters/interface"},
6416     ],
6417     "name": "body",
6418     "in": "body",
6419     "required": true,
6420     "schema": { "$ref": "#/definitions/Amacl" },
6421     "x-example":
6422     {
6423         "resources": [
6424             {
6425                 "href": "/temp",
6426                 "rt": ["oic.r.temperature"],
6427                 "if": ["oic.if.baseline", "oic.if.a"]
6428             },
6429             {
6430                 "href": "/temp",
6431                 "rt": ["oic.r.temperature"],
6432                 "if": ["oic.if.baseline", "oic.if.s"]
6433             }
6434         ]
6435     }
6436 },
6437 ],
6438 "responses": {
6439     "400": {
6440         "description": "The request is invalid."

```

```

6441         },
6442         "201": {
6443             "description": "The AMACL entry is created."
6444         }
6445     },
6446 },
6447 "delete": {
6448     "description": "Deletes the amacl data.\nWhen DELETE is used without query parameters, the
6449 entire collection is deleted.\nWhen DELETE uses the search parameter with \"subject\", only the
6450 matched entry is deleted.\n",
6451     "parameters": [
6452         {"$ref": "#/parameters/interface"},
6453         {
6454             "in": "query",
6455             "description": "Delete the ACE identified by the string matching the subject value.\n",
6456             "type": "string",
6457             "name": "subject"
6458         }
6459     ],
6460     "responses": {
6461         "200": {
6462             "description": "The ACE instance or the the entire AMACL resource has been
6463 successfully deleted."
6464         },
6465         "400": {
6466             "description": "The request is invalid."
6467         }
6468     }
6469 }
6470 },
6471 },
6472 "parameters": {
6473     "interface" : {
6474         "in" : "query",
6475         "name" : "if",
6476         "type" : "string",
6477         "enum" : ["oic.if.baseline"]
6478     }
6479 },
6480 "definitions": {
6481     "Amacl" : {
6482         "properties": {
6483             "rt" : {
6484                 "description": "Resource Type of the Resource",
6485                 "items": {
6486                     "maxLength": 64,
6487                     "type": "string",
6488                     "enum": ["oic.r.amacl"]
6489                 },
6490                 "minItems": 1,
6491                 "maxItems": 1,
6492                 "readOnly": true,
6493                 "type": "array"
6494             },
6495             "n": {
6496                 "$ref":
6497 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6498 schema.json#/definitions/n"
6499             },
6500             "id": {
6501                 "$ref":
6502 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6503 schema.json#/definitions/id"
6504             },
6505             "resources" : {
6506                 "description": "Multiple links to this host's resources",
6507                 "items": {
6508                     "description": "Each resource must have at least one of these properties set",
6509                     "properties": {
6510                         "href": {
6511                             "description": "When present, the ACE only applies when the href matches\nThis is

```

```

6512 the target URI, it can be specified as a Relative Reference or fully-qualified URI.",
6513     "format": "uri",
6514     "maxLength": 256,
6515     "type": "string"
6516 },
6517 "if": {
6518     "description": "When present, the ACE only applies when the if (interface)
6519 matches\nThe interface set supported by this resource",
6520     "items": {
6521         "enum": [
6522             "oic.if.baseline",
6523             "oic.if.ll",
6524             "oic.if.b",
6525             "oic.if.rw",
6526             "oic.if.r",
6527             "oic.if.a",
6528             "oic.if.s"
6529         ],
6530         "type": "string"
6531     },
6532     "minItems": 1,
6533     "type": "array"
6534 },
6535 "rt": {
6536     "description": "When present, the ACE only applies when the rt (resource type)
6537 matches\nResource Type of the Resource",
6538     "items": {
6539         "maxLength": 64,
6540         "type": "string"
6541     },
6542     "minItems": 1,
6543     "type": "array"
6544 },
6545 "wc": {
6546     "description": "A wildcard matching policy",
6547     "pattern": "^[~+*]$",
6548     "type": "string"
6549 }
6550 },
6551 "type": "object"
6552 },
6553 "type": "array"
6554 },
6555 "if" : {
6556     "description": "The interface set supported by this resource",
6557     "items": {
6558         "enum": [
6559             "oic.if.baseline"
6560         ],
6561         "type": "string"
6562     },
6563     "minItems": 1,
6564     "maxItems": 1,
6565     "readOnly": true,
6566     "type": "array"
6567 }
6568 },
6569 "type" : "object",
6570 "required": ["resources"]
6571 }
6572 }
6573 }
6574

```

6575 C.5.5 Property definition

6576 Table C.8 defines the Properties that are part of the ['oic.r.amacl'] Resource Type

6577 **Table C.8 – The Property definitions of the Resource with type 'rt' = ['oic.r.amacl']**

Property name	Value type	Mandatory	Access mode	Description
---------------	------------	-----------	-------------	-------------

n	multiple types: see schema	No	Read Write	
resources	array: see schema	Yes	Read Write	Multiple links to this host's resources
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource

6578 **C.5.6 CRUDN behaviour**

6579 Table C.9 defines the CRUDN operations that are supported on the ['oic.r.amacl'] Resource Type

6580 **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

6581 **C.6 Credential**

6582 **C.6.1 Introduction**

6583 This resource specifies credentials a device may use to establish secure communication.

6584 Retrieves the credential data.

6585 When used without query parameters, all the credential entries are returned.

6586 When used with a query parameter, only the credentials matching the specified
6587 parameter are returned.

6588

6589 Note that write-only credential data will not be returned.

6590

6591 **C.6.2 Well-known URI**

6592 /oic/sec/cred

6593 **C.6.3 Resource type**

6594 The resource type (rt) is defined as: ['oic.r.cred'].

6595 **C.6.4 OpenAPI 2.0 definition**

```

6596 {
6597   "swagger": "2.0",
6598   "info": {
6599     "title": "Credential",
6600     "version": "v1.0-20181031",
6601     "license": {
6602       "name": "OCF Data Model License",
6603       "url":
6604         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
6605         CENSE.md",
6606       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
6607         reserved."
6608     },
6609     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
6610   },
6611   "schemes": ["http"],
6612   "consumes": ["application/json"],
6613   "produces": ["application/json"],
6614   "paths": {
6615     "/oic/sec/cred" : {

```

```

6616     "get": {
6617         "description": "This resource specifies credentials a device may use to establish secure
6618 communication.\nRetrieves the credential data.\nWhen used without query parameters, all the
6619 credential entries are returned.\nWhen used with a query parameter, only the credentials matching
6620 the specified\nparameter are returned.\n\nNote that write-only credential data will not be
6621 returned.\n",
6622         "parameters": [
6623             {"$ref": "#/parameters/interface"}
6624             ,{"$ref": "#/parameters/cred-filtered-credid"}
6625             ,{"$ref": "#/parameters/cred-filtered-subjectuuid"}
6626         ],
6627         "responses": {
6628             "200": {
6629                 "description": "",
6630                 "x-example":
6631                 {
6632                     "rt": ["oic.r.cred"],
6633                     "creds": [
6634                         {
6635                             "credid": 55,
6636                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6637                             "roleid": {
6638                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6639                                 "role": "SOME_STRING"
6640                             },
6641                             "credtype": 32,
6642                             "publicdata": {
6643                                 "encoding": "oic.sec.encoding.base64",
6644                                 "data": "BASE-64-ENCODED-VALUE"
6645                             },
6646                             "privatedata": {
6647                                 "encoding": "oic.sec.encoding.base64",
6648                                 "data": "BASE-64-ENCODED-VALUE",
6649                                 "handle": 4
6650                             },
6651                             "optionaldata": {
6652                                 "revstat": false,
6653                                 "encoding": "oic.sec.encoding.base64",
6654                                 "data": "BASE-64-ENCODED-VALUE"
6655                             },
6656                             "period": "20160101T180000Z/20170102T070000Z",
6657                             "crms": [ "oic.sec.crm.pk10" ]
6658                         },
6659                         {
6660                             "credid": 56,
6661                             "subjectuuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6662                             "roleid": {
6663                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6664                                 "role": "SOME_STRING"
6665                             },
6666                             "credtype": 1,
6667                             "publicdata": {
6668                                 "encoding": "oic.sec.encoding.base64",
6669                                 "data": "BASE-64-ENCODED-VALUE"
6670                             },
6671                             "privatedata": {
6672                                 "encoding": "oic.sec.encoding.base64",
6673                                 "data": "BASE-64-ENCODED-VALUE",
6674                                 "handle": 4
6675                             },
6676                             "optionaldata": {
6677                                 "revstat": false,
6678                                 "encoding": "oic.sec.encoding.base64",
6679                                 "data": "BASE-64-ENCODED-VALUE"
6680                             },
6681                             "period": "20160101T180000Z/20170102T070000Z",
6682                             "crms": [ "oic.sec.crm.pk10" ]
6683                         }
6684                     ],
6685                     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6686                 }
6687             }
6688         }
6689     }

```

```

6687         ,
6688         "schema": { "$ref": "#/definitions/Cred" }
6689     },
6690     "400": {
6691         "description": "The request is invalid."
6692     }
6693 },
6694 },
6695 "post": {
6696     "description": "Updates the credential resource with the provided
6697 credentials.\n\nCredentials provided in the update with credid(s) not currently in the\ncredential
6698 resource are added.\n\nCredentials provided in the update with credid(s) already in the\ncredential
6699 resource completely replace the creds in the credential\nresource.\n\nCredentials provided in the
6700 update without credid(s) properties are\nadded and assigned unique credid(s) in the credential
6701 resource.\n",
6702     "parameters": [
6703         { "$ref": "#/parameters/interface" },
6704         {
6705             "name": "body",
6706             "in": "body",
6707             "required": true,
6708             "schema": { "$ref": "#/definitions/Cred-Update" },
6709             "x-example":
6710                 {
6711                     "creds": [
6712                         {
6713                             "credid": 55,
6714                             "subjectuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6715                             "roleid": {
6716                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6717                                 "role": "SOME_STRING"
6718                             },
6719                             "credtype": 32,
6720                             "publicdata": {
6721                                 "encoding": "oic.sec.encoding.base64",
6722                                 "data": "BASE-64-ENCODED-VALUE"
6723                             },
6724                             "privatedata": {
6725                                 "encoding": "oic.sec.encoding.base64",
6726                                 "data": "BASE-64-ENCODED-VALUE",
6727                                 "handle": 4
6728                             },
6729                             "optionaldata": {
6730                                 "revstat": false,
6731                                 "encoding": "oic.sec.encoding.base64",
6732                                 "data": "BASE-64-ENCODED-VALUE"
6733                             },
6734                             "period": "20160101T180000Z/20170102T070000Z",
6735                             "crms": [ "oic.sec.crm.pk10" ]
6736                         },
6737                         {
6738                             "credid": 56,
6739                             "subjectuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
6740                             "roleid": {
6741                                 "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
6742                                 "role": "SOME_STRING"
6743                             },
6744                             "credtype": 1,
6745                             "publicdata": {
6746                                 "encoding": "oic.sec.encoding.base64",
6747                                 "data": "BASE-64-ENCODED-VALUE"
6748                             },
6749                             "privatedata": {
6750                                 "encoding": "oic.sec.encoding.base64",
6751                                 "data": "BASE-64-ENCODED-VALUE",
6752                                 "handle": 4
6753                             },
6754                             "optionaldata": {
6755                                 "revstat": false,
6756                                 "encoding": "oic.sec.encoding.base64",
6757                                 "data": "BASE-64-ENCODED-VALUE"

```

```

6758         },
6759         "period": "20160101T180000Z/20170102T070000Z",
6760         "crms": [ "oic.sec.crm.pk10" ]
6761     }
6762 ],
6763     "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6764 }
6765 }
6766 ],
6767 "responses": {
6768     "400": {
6769         "description": "The request is invalid."
6770     },
6771     "201": {
6772         "description": "The credential entry is created."
6773     },
6774     "204": {
6775         "description": "The credential entry is updated."
6776     }
6777 }
6778 },
6779 "delete": {
6780     "description": "Deletes credential entries.\nWhen DELETE is used without query parameters,
6781 all the cred entries are deleted.\nWhen DELETE is used with a query parameter, only the entries
6782 matching\nthe query parameter are deleted.\n",
6783     "parameters": [
6784         {"$ref": "#/parameters/interface"},
6785         {"$ref": "#/parameters/cred-filtered-credid"},
6786         {"$ref": "#/parameters/cred-filtered-subjectuuid"}
6787     ],
6788     "responses": {
6789         "400": {
6790             "description": "The request is invalid."
6791         },
6792         "204": {
6793             "description": "The specific credential(s) or the the entire credential resource has
6794 been successfully deleted."
6795         }
6796     }
6797 }
6798 }
6799 },
6800 "parameters": {
6801     "interface": {
6802         "in": "query",
6803         "name": "if",
6804         "type": "string",
6805         "enum": ["oic.if.baseline"]
6806     },
6807     "cred-filtered-credid": {
6808         "in": "query",
6809         "name": "credid",
6810         "required": false,
6811         "type": "integer",
6812         "description": "Only applies to the credential with the specified credid",
6813         "x-example": 2112
6814     },
6815     "cred-filtered-subjectuuid": {
6816         "in": "query",
6817         "name": "subjectuuid",
6818         "required": false,
6819         "type": "string",
6820         "description": "Only applies to credentials with the specified subject UUID",
6821         "x-example": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
6822     }
6823 },
6824 "definitions": {
6825     "Cred": {
6826         "properties": {
6827             "rowneruuid": {
6828                 "description": "Format pattern according to IETF RFC 4122.",

```

```

6829         "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}$",
6830         "type": "string"
6831     },
6832     "rt" : {
6833         "description": "Resource Type of the Resource",
6834         "items": {
6835             "maxLength": 64,
6836             "type": "string",
6837             "enum": ["oic.r.cred"]
6838         },
6839         "minItems": 1,
6840         "readOnly": true,
6841         "type": "array",
6842         "uniqueItems": true
6843     },
6844     "n": {
6845         "$ref":
6846         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6847         schema.json#/definitions/n"
6848     },
6849     "id": {
6850         "$ref":
6851         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
6852         schema.json#/definitions/id"
6853     },
6854     "creds" : {
6855         "description": "List of credentials available at this resource",
6856         "items": {
6857             "properties": {
6858                 "credid": {
6859                     "description": "Local reference to a credential resource",
6860                     "type": "integer"
6861                 },
6862                 "credtype": {
6863                     "description": "Representation of this credential's type\nCredential Types - Cred
6864 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
6865 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
6866 password32 - Asymmetric encryption key",
6867                     "maximum": 63,
6868                     "minimum": 0,
6869                     "type": "integer"
6870                 },
6871                 "credusage": {
6872                     "description": "A string that provides hints about how/where the cred is used\nThe
6873 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
6874 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
6875 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
6876                     "enum": [
6877                         "oic.sec.cred.trustca",
6878                         "oic.sec.cred.cert",
6879                         "oic.sec.cred.rolecert",
6880                         "oic.sec.cred.mfgtrustca",
6881                         "oic.sec.cred.mfgcert"
6882                     ],
6883                     "type": "string"
6884                 },
6885                 "crms": {
6886                     "description": "The refresh methods that may be used to update this credential",
6887                     "items": {
6888                         "description": "Each enum represents a method by which the credentials are
6889 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
6890 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
6891 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
6892 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
6893                         "enum": [
6894                             "oic.sec.crm.pro",
6895                             "oic.sec.crm.psk",
6896                             "oic.sec.crm.rdp",
6897                             "oic.sec.crm.skdc",
6898                             "oic.sec.crm.pk10"
6899                         ]
6900                     }
6901                 }
6902             }
6903         }
6904     }

```

```

6900         "type": "string"
6901     },
6902     "type": "array",
6903     "uniqueItems" : true
6904 },
6905 "optionaldata": {
6906     "description": "Credential revocation status information\nOptional credential
6907 contents describes revocation status for this credential",
6908     "properties": {
6909         "data": {
6910             "description": "The encoded structure",
6911             "type": "string"
6912         },
6913         "encoding": {
6914             "description": "A string specifying the encoding format of the data contained in
6915 the optdata",
6916             "x-detail-desc": [
6917                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
6918                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
6919                 "oic.sec.encoding.base64 - Base64 encoded object",
6920                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
6921                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
6922                 "oic.sec.encoding.raw - Raw hex encoded data"
6923             ],
6924             "enum": [
6925                 "oic.sec.encoding.jwt",
6926                 "oic.sec.encoding.cwt",
6927                 "oic.sec.encoding.base64",
6928                 "oic.sec.encoding.pem",
6929                 "oic.sec.encoding.der",
6930                 "oic.sec.encoding.raw"
6931             ],
6932             "type": "string"
6933         },
6934         "revstat": {
6935             "description": "Revocation status flag - true = revoked",
6936             "type": "boolean"
6937         }
6938     },
6939     "required": [
6940         "revstat"
6941     ],
6942     "type": "object"
6943 },
6944 "period": {
6945     "description": "String with IETF RFC 5545 Period",
6946     "type": "string"
6947 },
6948 "privatedata": {
6949     "description": "Private credential information\nCredential resource non-public
6950 contents",
6951     "properties": {
6952         "data": {
6953             "description": "The encoded value",
6954             "maxLength": 3072,
6955             "type": "string"
6956         },
6957         "encoding": {
6958             "description": "A string specifying the encoding format of the data contained in
6959 the privdata\noic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT)
6960 encoding\noic.sec.encoding.cwt - IETF RFC 8392 encoding\noic.sec.encoding.base64 - Base64 encoded
6961 object\noic.sec.encoding.uri - URI reference\noic.sec.encoding.handle - Data is contained in a
6962 storage sub-system referenced using a handle\noic.sec.encoding.raw - Raw hex encoded data",
6963             "enum": [
6964                 "oic.sec.encoding.jwt",
6965                 "oic.sec.encoding.cwt",
6966                 "oic.sec.encoding.base64",
6967                 "oic.sec.encoding.uri",
6968                 "oic.sec.encoding.handle",
6969                 "oic.sec.encoding.raw"
6970             ],

```

```

6971         "type": "string"
6972     },
6973     "handle": {
6974         "description": "Handle to a key storage resource",
6975         "type": "integer"
6976     }
6977 },
6978 "required": [
6979     "encoding"
6980 ],
6981 "type": "object"
6982 },
6983 "publicdata": {
6984     "description": "Public credential information",
6985     "properties": {
6986         "data": {
6987             "description": "The encoded value",
6988             "maxLength": 3072,
6989             "type": "string"
6990         },
6991         "encoding": {
6992             "description": "A string specifying the encoding format of the data contained in
6993 the pubdata\noic.sec.encoding.jwt - IETF RFC7519 JSON web token (JWT) encoding\noic.sec.encoding.cwt
6994 - IETF RFC 8392 encoding\noic.sec.encoding.base64 - Base64 encoded object\noic.sec.encoding.uri -
6995 URI reference\noic.sec.encoding.pem - Encoding for PEM encoded certificate or
6996 chain\noic.sec.encoding.der - Encoding for DER encoded certificate\noic.sec.encoding.raw - Raw hex
6997 encoded data",
6998             "enum": [
6999                 "oic.sec.encoding.jwt",
7000                 "oic.sec.encoding.cwt",
7001                 "oic.sec.encoding.base64",
7002                 "oic.sec.encoding.uri",
7003                 "oic.sec.encoding.pem",
7004                 "oic.sec.encoding.der",
7005                 "oic.sec.encoding.raw"
7006             ],
7007             "type": "string"
7008         }
7009     },
7010     "type": "object"
7011 },
7012 "roleid": {
7013     "description": "The role this credential possesses\nSecurity role specified as an
7014 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
7015     "properties": {
7016         "authority": {
7017             "description": "The Authority component of the entity being identified. A NULL
7018 <Authority> refers to the local entity or device.",
7019             "type": "string"
7020         },
7021         "role": {
7022             "description": "The ID of the role being identified.",
7023             "type": "string"
7024         }
7025     },
7026     "required": [
7027         "role"
7028     ],
7029     "type": "object"
7030 },
7031 "subjectuuid": {
7032     "anyOf": [
7033         {
7034             "description": "The id of the device, which the cred entry applies to or \"*\
7035 for wildcard identity",
7036             "pattern": "^[\\*$]",
7037             "type": "string"
7038         },
7039         {
7040             "description": "Format pattern according to IETF RFC 4122.",
7041             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-

```

```

7042 F0-9]{12}$",
7043         "type": "string"
7044     }
7045 ]
7046 }
7047 },
7048 "type": "object"
7049 },
7050 "type": "array"
7051 },
7052 "if" : {
7053     "description": "The interface set supported by this resource",
7054     "items": {
7055         "enum": [
7056             "oic.if.baseline"
7057         ],
7058         "type": "string"
7059     },
7060     "minItems": 1,
7061     "readOnly": true,
7062     "type": "array"
7063 }
7064 },
7065 "type" : "object",
7066 "required": ["creds", "rowneruuid"]
7067 },
7068 "Cred-Update" : {
7069     "properties": {
7070         "rowneruuid" : {
7071             "description": "Format pattern according to IETF RFC 4122.",
7072             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7073             "type": "string"
7074         },
7075         "creds" : {
7076             "description": "List of credentials available at this resource",
7077             "items": {
7078                 "properties": {
7079                     "credid": {
7080                         "description": "Local reference to a credential resource",
7081                         "type": "integer"
7082                     },
7083                     "credtype": {
7084                         "description": "Representation of this credential's type\nCredential Types - Cred
7085 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7086 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
7087 password32 - Asymmetric encryption key",
7088                         "maximum": 63,
7089                         "minimum": 0,
7090                         "type": "integer"
7091                     },
7092                     "credusage": {
7093                         "description": "A string that provides hints about how/where the cred is used\nThe
7094 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
7095 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
7096 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
7097                         "enum": [
7098                             "oic.sec.cred.trustca",
7099                             "oic.sec.cred.cert",
7100                             "oic.sec.cred.rolecert",
7101                             "oic.sec.cred.mfgtrustca",
7102                             "oic.sec.cred.mfgcert"
7103                         ],
7104                         "type": "string"
7105                     },
7106                     "crms": {
7107                         "description": "The refresh methods that may be used to update this credential",
7108                         "items": {
7109                             "description": "Each enum represents a method by which the credentials are
7110 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
7111 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
7112 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution

```

```

7113 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
7114     "enum": [
7115         "oic.sec.crm.pro",
7116         "oic.sec.crm.psk",
7117         "oic.sec.crm.rdp",
7118         "oic.sec.crm.skdc",
7119         "oic.sec.crm.pk10"
7120     ],
7121     "type": "string"
7122 },
7123     "type": "array"
7124 },
7125     "optionaldata": {
7126         "description": "Credential revocation status information\nOptional credential
7127 contents describes revocation status for this credential",
7128         "properties": {
7129             "data": {
7130                 "description": "The encoded structure",
7131                 "type": "string"
7132             },
7133             "encoding": {
7134                 "description": "A string specifying the encoding format of the data contained in
7135 the optdata",
7136                 "x-detail-desc": [
7137                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
7138                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7139                     "oic.sec.encoding.base64 - Base64 encoded object",
7140                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
7141                     "oic.sec.encoding.der - Encoding for DER encoded certificate",
7142                     "oic.sec.encoding.raw - Raw hex encoded data"
7143                 ],
7144                 "enum": [
7145                     "oic.sec.encoding.jwt",
7146                     "oic.sec.encoding.cwt",
7147                     "oic.sec.encoding.base64",
7148                     "oic.sec.encoding.pem",
7149                     "oic.sec.encoding.der",
7150                     "oic.sec.encoding.raw"
7151                 ],
7152                 "type": "string"
7153             },
7154             "revstat": {
7155                 "description": "Revocation status flag - true = revoked",
7156                 "type": "boolean"
7157             }
7158         },
7159         "required": [
7160             "revstat"
7161         ],
7162         "type": "object"
7163     },
7164     "period": {
7165         "description": "String with IETF RFC 5545 Period",
7166         "type": "string"
7167     },
7168     "privatedata": {
7169         "description": "Private credential information\nCredential resource non-public
7170 contents",
7171         "properties": {
7172             "data": {
7173                 "description": "The encoded value",
7174                 "maxLength": 3072,
7175                 "type": "string"
7176             },
7177             "encoding": {
7178                 "description": "A string specifying the encoding format of the data contained in
7179 the privdata",
7180                 "x-detail-desc": [
7181                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
7182                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7183                     "oic.sec.encoding.base64 - Base64 encoded object",

```

```

7184         "oic.sec.encoding.uri - URI reference",
7185         "oic.sec.encoding.handle - Data is contained in a storage sub-system
referenced using a handle",
7186         "oic.sec.encoding.raw - Raw hex encoded data"
7187     ],
7188     },
7189     "enum": [
7190         "oic.sec.encoding.jwt",
7191         "oic.sec.encoding.cwt",
7192         "oic.sec.encoding.base64",
7193         "oic.sec.encoding.uri",
7194         "oic.sec.encoding.handle",
7195         "oic.sec.encoding.raw"
7196     ],
7197     "type": "string"
7198 },
7199 "handle": {
7200     "description": "Handle to a key storage resource",
7201     "type": "integer"
7202 },
7203 },
7204 "required": [
7205     "encoding"
7206 ],
7207 "type": "object"
7208 },
7209 "publicdata": {
7210     "properties": {
7211         "data": {
7212             "description": "The encoded value",
7213             "maxLength": 3072,
7214             "type": "string"
7215         },
7216         "encoding": {
7217             "description": "Public credential information\nA string specifying the encoding
format of the data contained in the pubdata",
7218             "x-detail-desc": [
7219                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
7220                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
7221                 "oic.sec.encoding.base64 - Base64 encoded object",
7222                 "oic.sec.encoding.uri - URI reference",
7223                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
7224                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
7225                 "oic.sec.encoding.raw - Raw hex encoded data"
7226             ],
7227             "enum": [
7228                 "oic.sec.encoding.jwt",
7229                 "oic.sec.encoding.cwt",
7230                 "oic.sec.encoding.base64",
7231                 "oic.sec.encoding.uri",
7232                 "oic.sec.encoding.pem",
7233                 "oic.sec.encoding.der",
7234                 "oic.sec.encoding.raw"
7235             ],
7236             "type": "string"
7237         }
7238     }
7239 },
7240 "type": "object"
7241 },
7242 "roleid": {
7243     "description": "The role this credential possesses\nSecurity role specified as an
<Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
7244     "properties": {
7245         "authority": {
7246             "description": "The Authority component of the entity being identified. A NULL
<Authority> refers to the local entity or device.",
7247             "type": "string"
7248         },
7249         "role": {
7250             "description": "The ID of the role being identified.",
7251             "type": "string"
7252         }
7253     }
7254 }

```

```

7255     },
7256     "required": [
7257         "role"
7258     ],
7259     "type": "object"
7260 },
7261 "subjectuuid": {
7262     "anyOf": [
7263         {
7264             "description": "The id of the device, which the cred entry applies to or \"*\
7265 for wildcard identity",
7266             "pattern": "^\\*$",
7267             "type": "string"
7268         },
7269         {
7270             "description": "Format pattern according to IETF RFC 4122.",
7271             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
7272 F0-9]{12}$",
7273             "type": "string"
7274         }
7275     ]
7276 },
7277 },
7278 "type": "object"
7279 },
7280 "type": "array"
7281 },
7282 "if" :
7283 {
7284     "description": "The interface set supported by this resource",
7285     "items": {
7286         "enum": [
7287             "oic.if.baseline"
7288         ],
7289         "type": "string"
7290     },
7291     "minItems": 1,
7292     "readOnly": true,
7293     "type": "array"
7294 }
7295 },
7296 "type" : "object"
7297 }
7298 }
7299 }
7300

```

7301 C.6.5 Property definition

7302 Table C.10 defines the Properties that are part of the ['oic.r.cred'] Resource Type

7303 **Table C.10 – The Property definitions of the Resource with type 'rt' = ['oic.r.cred']**

Property name	Value type	Mandatory	Access mode	Description
owneruuid	string	No	Read Write	Format pattern according to IETF RFC 4122.
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
creds	array: see schema	No	Read Write	List of credentials available at this resource
creds	array: see schema	Yes	Read Write	List of credentials

				available at this resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
n	multiple types: see schema	No	Read Write	

7304 **C.6.6 CRUDN behaviour**

7305 Table C.11 defines the CRUDN operations that are supported on the ['oic.r.cred'] Resource Type

7306 **Table C.11 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.cred']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

7307 **C.7 Certificate Revocation**

7308 **C.7.1 Introduction**

7309 This resource specifies certificate revocation lists as X.509 objects.

7310

7311 **C.7.2 Well-known URI**

7312 /oic/sec/crl

7313 **C.7.3 Resource type**

7314 The resource type (rt) is defined as: ['oic.r.crl'].

7315 **C.7.4 OpenAPI 2.0 definition**

```

7316 {
7317   "swagger": "2.0",
7318   "info": {
7319     "title": "Certificate Revocation",
7320     "version": "v1.0-20150819",
7321     "license": {
7322       "name": "OCF Data Model License",
7323       "url":
7324         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7325         CENSE.md",
7326       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7327         reserved."
7328     },
7329     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7330   },
7331   "schemes": ["http"],
7332   "consumes": ["application/json"],
7333   "produces": ["application/json"],
7334   "paths": {
7335     "/oic/sec/crl" : {
7336       "get": {
7337         "description": "This resource specifies certificate revocation lists as X.509 objects.\n",
7338         "parameters": [
7339           {"$ref": "#/parameters/interface"}
7340         ]
7341       }
7342     }
7343   }

```

```

7341     "responses": {
7342         "200": {
7343             "description": "",
7344             "x-example":
7345                 {
7346                     "rt": ["oic.r.crl"],
7347                     "crlid": 1,
7348                     "thisupdate": "2016-04-12T23:20:50.52Z",
7349                     "crldata": "Base64ENCODEDCRL"
7350                 },
7351             "schema": { "$ref": "#/definitions/Crl" }
7352         }
7353     },
7354 },
7355 "post": {
7356     "description": "Updates the CRL data\n",
7357     "parameters": [
7358         { "$ref": "#/parameters/interface" },
7359         {
7360             "name": "body",
7361             "in": "body",
7362             "required": true,
7363             "schema": { "$ref": "#/definitions/Crl-Update" },
7364             "x-example":
7365                 {
7366                     "crlid": 1,
7367                     "thisupdate": "2016-04-12T23:20:50.52Z",
7368                     "crldata": "Base64ENCODEDCRL"
7369                 }
7370         }
7371     ],
7372     "responses": {
7373         "400": {
7374             "description": "The request is invalid."
7375         },
7376         "204": {
7377             "description": "The CRL entry is updated."
7378         }
7379     }
7380 },
7381 },
7382 },
7383 "parameters": {
7384     "interface": {
7385         "in": "query",
7386         "name": "if",
7387         "type": "string",
7388         "enum": ["oic.if.baseline"]
7389     }
7390 },
7391 "definitions": {
7392     "Crl": {
7393         "properties": {
7394             "rt": {
7395                 "description": "Resource Type of the Resource",
7396                 "items": {
7397                     "maxLength": 64,
7398                     "type": "string",
7399                     "enum": ["oic.r.crl"]
7400                 },
7401                 "minItems": 1,
7402                 "readOnly": true,
7403                 "type": "array"
7404             },
7405             "n": {
7406                 "$ref":
7407 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7408 schema.json#/definitions/n"
7409             },
7410             "id": {
7411                 "$ref":

```

```

7412 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7413 schema.json#/definitions/id"
7414 },
7415 "crldata" : {
7416   "description": "Base64 BER encoded crl data",
7417   "type": "string"
7418 },
7419 "crlid" : {
7420   "description": "Local reference to a crl resource",
7421   "type": "integer"
7422 },
7423 "thisupdate" : {
7424   "description": "UTC time of last CRL update",
7425   "type": "string"
7426 },
7427 "if" : {
7428   "description": "The interface set supported by this resource",
7429   "items": {
7430     "enum": [
7431       "oic.if.baseline"
7432     ],
7433     "type": "string"
7434   },
7435   "minItems": 1,
7436   "readOnly": true,
7437   "type": "array"
7438 }
7439 },
7440 "type": "object",
7441 "required": ["crlid", "thisupdate", "crldata"]
7442 }
7443 ,
7444 "Crl-Update": {
7445   "properties": {
7446     "crldata": {
7447       "description": "Base64 BER encoded crl data",
7448       "type": "string"
7449     },
7450     "crlid": {
7451       "description": "Local reference to a crl resource",
7452       "type": "integer"
7453     },
7454     "thisupdate": {
7455       "description": "UTC time of last CRL update",
7456       "type": "string"
7457     }
7458   },
7459   "type" : "object"
7460 }
7461 }
7462 }
7463

```

7464 C.7.5 Property definition

7465 Table C.12 defines the Properties that are part of the ['oic.r.crl'] Resource Type

7466 **Table C.12 – The Property definitions of the Resource with type 'rt' = ['oic.r.crl']**

Property name	Value type	Mandatory	Access mode	Description
thisupdate	string	Yes	Read Write	UTC time of last CRL update
id	multiple types: see schema	No	Read Write	
n	multiple types: see schema	No	Read Write	
if	array: see schema	No	Read Only	The OCF Interface set

				supported by this resource
rt	array: schema see	No	Read Only	Resource Type of the Resource
crlid	integer	Yes	Read Write	Local reference to a crl resource
crldata	string	Yes	Read Write	Base64 BER encoded crl data
thisupdate	string		Read Write	UTC time of last CRL update
crlid	integer		Read Write	Local reference to a crl resource
crldata	string		Read Write	Base64 BER encoded crl data

7467 **C.7.6 CRUDN behaviour**

7468 Table C.13 defines the CRUDN operations that are supported on the ['oic.r.crl'] Resource Type

7469 **Table C.13 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.crl']**

Create	Read	Update	Delete	Notify
	get	post		observe

7470 **C.8 Certificate Signing Request**

7471 **C.8.1 Introduction**

7472 This resource specifies a Certificate Signing Request.

7473

7474 **C.8.2 Well-known URI**

7475 /oic/sec/csr

7476 **C.8.3 Resource type**

7477 The resource type (rt) is defined as: ['oic.r.csr'].

7478 **C.8.4 OpenAPI 2.0 definition**

```

7479 {
7480   "swagger": "2.0",
7481   "info": {
7482     "title": "Certificate Signing Request",
7483     "version": "v1.0-20150819",
7484     "license": {
7485       "name": "OCF Data Model License",
7486       "url":
7487         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7488         CENSE.md",
7489       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7490         reserved."
7491     },
7492     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7493   },
7494   "schemes": ["http"],
7495   "consumes": ["application/json"],
7496   "produces": ["application/json"],
7497   "paths": {
7498     "/oic/sec/csr" : {
7499       "get": {
7500         "description": "This resource specifies a Certificate Signing Request.\n",
7501         "parameters": [
7502           {"$ref": "#/parameters/interface"}
7503         ],
7504         "responses": {

```

```

7505         "200": {
7506             "description": "",
7507             "x-example":
7508                 {
7509                     "rt": ["oic.r.csr"],
7510                     "encoding": "oic.sec.encoding.pem",
7511                     "csr": "PEMENCODEDCSR"
7512                 },
7513             "schema": { "$ref": "#/definitions/Csr" }
7514         },
7515         "404": {
7516             "description": "The device does not support certificates and generating CSRs."
7517         },
7518         "503": {
7519             "description": "The device is not yet ready to return a response. Try again later."
7520         }
7521     }
7522 }
7523 },
7524 },
7525 "parameters": {
7526     "interface": {
7527         "in": "query",
7528         "name": "if",
7529         "type": "string",
7530         "enum": ["oic.if.baseline"]
7531     }
7532 },
7533 "definitions": {
7534     "Csr": {
7535         "properties": {
7536             "rt": {
7537                 "description": "Resource Type of the Resource",
7538                 "items": {
7539                     "maxLength": 64,
7540                     "type": "string",
7541                     "enum": ["oic.r.csr"]
7542                 },
7543                 "minItems": 1,
7544                 "readOnly": true,
7545                 "type": "array"
7546             },
7547             "encoding": {
7548                 "description": "A string specifying the encoding format of the data contained in csr",
7549                 "x-detail-desc": [
7550                     "oic.sec.encoding.pem - Encoding for PEM encoded CSR",
7551                     "oic.sec.encoding.der - Encoding for DER encoded CSR"
7552                 ],
7553                 "enum": [
7554                     "oic.sec.encoding.pem",
7555                     "oic.sec.encoding.der"
7556                 ],
7557                 "readOnly": true,
7558                 "type": "string"
7559             },
7560             "n": {
7561                 "$ref":
7562                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7563                 schema.json#/definitions/n"
7564             },
7565             "id": {
7566                 "$ref":
7567                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7568                 schema.json#/definitions/id"
7569             },
7570             "csr": {
7571                 "description": "Signed CSR in ASN.1 in the encoding specified by the encoding property",
7572                 "maxLength": 3072,
7573                 "readOnly": true,
7574                 "type": "string"
7575             }

```

```

7576     "if": {
7577         "description": "The interface set supported by this resource",
7578         "items": {
7579             "enum": [
7580                 "oic.if.baseline"
7581             ],
7582             "type": "string"
7583         },
7584         "minItems": 1,
7585         "readOnly": true,
7586         "type": "array"
7587     }
7588 },
7589 "type" : "object",
7590 "required": ["csr", "encoding"]
7591 }
7592 }
7593 }
7594

```

7595 C.8.5 Property definition

7596 Table C.14 defines the Properties that are part of the ['oic.r.csr'] Resource Type

7597 **Table C.14 – The Property definitions of the Resource with type 'rt' = ['oic.r.csr']**

Property name	Value type	Mandatory	Access mode	Description
csr	string	Yes	Read Only	Signed CSR in ASN.1 in the encoding specified by the encoding property
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
encoding	string	Yes	Read Only	A string specifying the encoding format of the data contained in csr
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource

7598 C.8.6 CRUDN behaviour

7599 Table C.15 defines the CRUDN operations that are supported on the ['oic.r.csr'] Resource Type

7600 **Table C.15 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.csr']**

Create	Read	Update	Delete	Notify
	get			observe

7601 C.9 Device Owner Transfer Method

7602 C.9.1 Introduction

7603 This resource specifies properties needed to establish a device owner.

7604

7605 **C.9.2 Well-known URI**

7606 /oic/sec/doxm

7607 **C.9.3 Resource type**

7608 The resource type (rt) is defined as: ['oic.r.doxm'].

7609 **C.9.4 OpenAPI 2.0 definition**

```
7610 {
7611   "swagger": "2.0",
7612   "info": {
7613     "title": "Device Owner Transfer Method",
7614     "version": "v1.0-20181001",
7615     "license": {
7616       "name": "OCF Data Model License",
7617       "url":
7618         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7619         CENSE.md",
7620       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7621 reserved."
7622     },
7623     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7624   },
7625   "schemes": ["http"],
7626   "consumes": ["application/json"],
7627   "produces": ["application/json"],
7628   "paths": {
7629     "/oic/sec/doxm" : {
7630       "get": {
7631         "description": "This resource specifies properties needed to establish a device owner.\n",
7632         "parameters": [
7633           {"$ref": "#/parameters/interface"}
7634         ],
7635         "responses": {
7636           "200": {
7637             "description": "",
7638             "x-example":
7639               {
7640                 "rt": ["oic.r.doxm"],
7641                 "oxms": [ 0, 2, 3 ],
7642                 "oxmsel": 0,
7643                 "sct": 16,
7644                 "owned": true,
7645                 "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7646                 "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
7647                 "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7648               }
7649           },
7650           "schema": { "$ref": "#/definitions/Doxm" }
7651         },
7652         "400": {
7653           "description": "The request is invalid."
7654         }
7655       }
7656     },
7657     "post": {
7658       "description": "Updates the DOXM resource data\n",
7659       "parameters": [
7660         {"$ref": "#/parameters/interface"},
7661         {
7662           "name": "body",
7663           "in": "body",
7664           "required": true,
7665           "schema": { "$ref": "#/definitions/Doxm-Update" },
7666           "x-example":
7667             {
7668               "oxmsel": 0,
7669               "owned": true,
7670               "deviceuuid": "de305d54-75b4-431b-adb2-eb6b9e546014",
7671               "devowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9",
```

```

7672         "rowneruuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"
7673     }
7674 }
7675 ],
7676 "responses": {
7677     "400": {
7678         "description": "The request is invalid."
7679     },
7680     "204": {
7681         "description": "The DOXM entry is updated."
7682     }
7683 }
7684 }
7685 }
7686 },
7687 "parameters": {
7688     "interface": {
7689         "in": "query",
7690         "name": "if",
7691         "type": "string",
7692         "enum": ["oic.if.baseline"]
7693     }
7694 },
7695 "definitions": {
7696     "Doxm": {
7697         "properties": {
7698             "rowneruuid": {
7699                 "description": "Format pattern according to IETF RFC 4122.",
7700                 "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}$",
7701                 "type": "string"
7702             },
7703             "oxms": {
7704                 "description": "List of supported owner transfer methods",
7705                 "items": {
7706                     "description": "The device owner transfer methods that may be selected at device on-
7707 boarding. Each value indicates a specific Owner Transfer method0 - Numeric OTM identifier for the
7708 Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method
7709 (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method
7710 (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method (oic.sec.doxm.dcap)
7711 (deprecated)",
7712                 "type": "integer"
7713             },
7714             "readOnly": true,
7715             "type": "array"
7716         },
7717         "devowneruuid": {
7718             "description": "Format pattern according to IETF RFC 4122.",
7719             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7720             "type": "string"
7721         },
7722         "deviceuuid": {
7723             "description": "The uuid formatted identity of the device\nFormat pattern according to
7724 IETF RFC 4122.",
7725             "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}$",
7726             "type": "string"
7727         },
7728         "owned": {
7729             "description": "Ownership status flag",
7730             "type": "boolean"
7731         },
7732         "n": {
7733             "$ref":
7734 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7735 schema.json#/definitions/n"
7736         },
7737         "id": {
7738             "$ref":
7739 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7740 schema.json#/definitions/id"
7741         },
7742         "oxmsel": {

```

```

7743         "description": "The selected owner transfer method used during on-boarding\nThe device
7744 owner transfer methods that may be selected at device on-boarding. Each value indicates a specific
7745 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7746 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7747 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7748 method (oic.sec.doxm.dcap) (deprecated)",
7749         "type": "integer"
7750     },
7751     "sct": {
7752         "description": "Bitmask encoding of supported credential types\nCredential Types -
7753 Cred type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
7754 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
7755 password32 - Asymmetric encryption key",
7756         "maximum": 63,
7757         "minimum": 0,
7758         "type": "integer",
7759         "readOnly": true
7760     },
7761     "rt": {
7762         "description": "Resource Type of the Resource",
7763         "items": {
7764             "maxLength": 64,
7765             "type": "string",
7766             "enum": ["oic.r.doxm"]
7767         },
7768         "minItems": 1,
7769         "readOnly": true,
7770         "type": "array"
7771     },
7772     "if": {
7773         "description": "The interface set supported by this resource",
7774         "items": {
7775             "enum": [
7776                 "oic.if.baseline"
7777             ],
7778             "type": "string"
7779         },
7780         "minItems": 1,
7781         "readOnly": true,
7782         "type": "array"
7783     }
7784 },
7785 "type": "object",
7786 "required": ["oxms", "oxmsel", "sct", "owned", "deviceuuid", "devowneruuid", "rowneruuid"]
7787 },
7788 "Doxm-Update": {
7789     "properties": {
7790         "rowneruuid": {
7791             "description": "Format pattern according to IETF RFC 4122.",
7792             "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}$",
7793             "type": "string"
7794         },
7795         "devowneruuid": {
7796             "description": "Format pattern according to IETF RFC 4122.",
7797             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7798             "type": "string"
7799         },
7800         "deviceuuid": {
7801             "description": "The uuid formatted identity of the device\nFormat pattern according to
7802 IETF RFC 4122.",
7803             "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
7804 9]{12}$",
7805             "type": "string"
7806         },
7807         "owned": {
7808             "description": "Ownership status flag",
7809             "type": "boolean"
7810         },
7811         "oxmsel": {
7812             "description": "The selected owner transfer method used during on-boarding\nThe device
7813 owner transfer methods that may be selected at device on-boarding. Each value indicates a specific

```

```

7814 Owner Transfer method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 -
7815 Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for
7816 the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap
7817 method (oic.sec.doxm.dcap) (deprecated)",
7818     "type": "integer"
7819     },
7820   },
7821   "type" : "object"
7822 }
7823 }
7824 }
7825 }

```

7826 C.9.5 Property definition

7827 Table C.16 defines the Properties that are part of the ['oic.r.doxm'] Resource Type

7828 **Table C.16 – The Property definitions of the Resource with type 'rt' = ['oic.r.doxm']**

Property name	Value type	Mandatory	Access mode	Description
devowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
oxms	array: see schema	Yes	Read Only	List of supported owner transfer methods
rt	array: see schema	No	Read Only	Resource Type of the Resource
owned	boolean	Yes	Read Write	Ownership status flag
rowneruuid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string	Yes	Read Write	The uuid formatted identity of the device Format pattern according to IETF RFC 4122.
oxmsel	integer	Yes	Read Write	The selected owner transfer method used during on-boarding The device owner transfer methods that may be selected at device on-boarding. Each value indicates a specific Owner Transfer method method0 - Numeric OTM identifier for the Just-Works method (oic.sec.doxm.jw)1 - Numeric OTM identifier for the random PIN method (oic.sec.doxm.rdp)2 - Numeric OTM identifier for the manufacturer certificate method (oic.sec.doxm.mfgcert)3 - Numeric OTM identifier for the decap method

				(oic.sec.doxm.dcap) (deprecated)
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
n	multiple types: see schema	No	Read Write	
sct	integer	Yes	Read Only	Bitmask encoding of supported credential types Credential Types - Cred type encoded as a bitmask. 0 - Empty credential used for 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
id	multiple types: see schema	No	Read Write	
rowneruuid	string		Read Write	Format pattern according to IETF RFC 4122.
deviceuuid	string		Read Write	The uuid formatted identity of the device Format pattern according to IETF RFC 4122.
owned	boolean		Read Write	Ownership status flag
oxmsel	integer		Read Write	The selected owner transfer method used during on-boarding The device owner transfer methods that may be selected at device on-boarding. Each value indicates a specific Owner Transfer method 0 - 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.

7829 **C.9.6 CRUDN behaviour**

7830 Table C.17 defines the CRUDN operations that are supported on the ['oic.r.doxm'] Resource Type

7831 **Table C.17 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.doxm']**

Create	Read	Update	Delete	Notify
	get	post		observe

7832 **C.10 Device Provisioning Status**

7833 **C.10.1 Introduction**

7834 This resource specifies device provisioning status.

7835

7836 **C.10.2 Well-known URI**

7837 /oic/sec/pstat

7838 **C.10.3 Resource type**

7839 The resource type (rt) is defined as: ['oic.r.pstat'].

7840 **C.10.4 OpenAPI 2.0 definition**

```

7841 {
7842   "swagger": "2.0",
7843   "info": {
7844     "title": "Device Provisioning Status",
7845     "version": "v1.0-20191001",
7846     "license": {
7847       "name": "OCF Data Model License",
7848       "url":
7849         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
7850         CENSE.md",
7851       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
7852         reserved."
7853     },
7854     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
7855   },
7856   "schemes": ["http"],
7857   "consumes": ["application/json"],
7858   "produces": ["application/json"],
7859   "paths": {
7860     "/oic/sec/pstat" : {
7861       "get": {
7862         "description": "This resource specifies device provisioning status.\n",
7863         "parameters": [
7864           {"$ref": "#/parameters/interface"}
7865         ],
7866         "responses": {
7867           "200": {
7868             "description": "",
7869             "x-example":
7870               {
7871                 "rt": ["oic.r.pstat"],
7872                 "dos": {"s": 3, "p": true},
7873                 "isop": true,

```

```

7874         "cm": 8,
7875         "tm": 60,
7876         "om": 2,
7877         "sm": 7,
7878         "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7879     },
7880     "schema": { "$ref": "#/definitions/Pstat" }
7881 },
7882 "400": {
7883     "description": "The request is invalid."
7884 }
7885 },
7886 },
7887 "post": {
7888     "description": "Sets or updates device provisioning status data.\n",
7889     "parameters": [
7890         { "$ref": "#/parameters/interface",
7891           {
7892             "name": "body",
7893             "in": "body",
7894             "required": true,
7895             "schema": { "$ref": "#/definitions/Pstat-Update" },
7896             "x-example":
7897               {
7898                 "dos": {"s": 3},
7899                 "tm": 60,
7900                 "om": 2,
7901                 "rowneruuid": "de305d54-75b4-431b-adb2-eb6b9e546014"
7902               }
7903           }
7904         ],
7905     "responses": {
7906         "400": {
7907             "description": "The request is invalid."
7908         },
7909         "204": {
7910             "description": "The PSTAT entry is updated."
7911         }
7912     }
7913 },
7914 },
7915 },
7916 "parameters": {
7917     "interface": {
7918         "in": "query",
7919         "name": "if",
7920         "type": "string",
7921         "enum": ["oic.if.baseline"]
7922     }
7923 },
7924 "definitions": {
7925     "Pstat": {
7926         "properties": {
7927             "rowneruuid": {
7928                 "description": "The UUID formatted identity of the Resource owner\nFormat pattern
7929 according to IETF RFC 4122.",
7930                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
7931                 "type": "string"
7932             },
7933             "rt": {
7934                 "description": "Resource Type of the Resource",
7935                 "items": {
7936                     "maxLength": 64,
7937                     "type": "string",
7938                     "enum": ["oic.r.pstat"]
7939                 },
7940                 "minItems": 1,
7941                 "readOnly": true,
7942                 "type": "array"
7943             },
7944         }

```

```

7945         "description": "Current operational mode\nDevice provisioning operation may be server
7946 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7947 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7948 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7949 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
7950         "maximum": 7,
7951         "minimum": 1,
7952         "type": "integer"
7953     },
7954     "cm": {
7955         "description": "Current device provisioning mode\nDevice provisioning mode maintains a
7956 bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
7957 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7958 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7959 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7960 Software Version Validation128 - Initiate Secure Software Update",
7961         "maximum": 255,
7962         "minimum": 0,
7963         "type": "integer",
7964         "readOnly": true
7965     },
7966     "n": {
7967         "$ref":
7968 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7969 schema.json#/definitions/n"
7970     },
7971     "id": {
7972         "$ref":
7973 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
7974 schema.json#/definitions/id"
7975     },
7976     "isop": {
7977         "description": "true indicates device is operational",
7978         "readOnly": true,
7979         "type": "boolean"
7980     },
7981     "tm": {
7982         "description": "Target device provisioning mode\nDevice provisioning mode maintains a
7983 bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
7984 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
7985 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
7986 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
7987 Software Version Validation128 - Initiate Secure Software Update",
7988         "maximum": 255,
7989         "minimum": 0,
7990         "type": "integer"
7991     },
7992     "sm": {
7993         "description": "Supported operational modes\nDevice provisioning operation may be server
7994 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
7995 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
7996 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
7997 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
7998         "maximum": 7,
7999         "minimum": 1,
8000         "type": "integer",
8001         "readOnly": true
8002     },
8003     "dos": {
8004         "description": "Device on-boarding state\nDevice operation state machine",
8005         "properties": {
8006             "p": {
8007                 "default": true,
8008                 "description": "'p' is TRUE when the 's' state is pending until all necessary changes
8009 to device resources are complete.",
8010                 "readOnly": true,
8011                 "type": "boolean"
8012             },
8013             "s": {
8014                 "description": "The current or pending operational state",
8015                 "x-detail-desc": [

```

```

8016         "0 - RESET - Device reset state",
8017         "1 - RFOFM - Ready for device owner transfer method state",
8018         "2 - RFPRO - Ready for device provisioning state",
8019         "3 - RFNOP - Ready for device normal operation state",
8020         "4 - SRESET - The device is in a soft reset state"
8021     ],
8022     "maximum": 4,
8023     "minimum": 0,
8024     "type": "integer"
8025 }
8026 },
8027 "required": [
8028     "s"
8029 ],
8030 "type": "object"
8031 },
8032 "if" : {
8033     "description": "The interface set supported by this resource",
8034     "items": {
8035         "enum": [
8036             "oic.if.baseline"
8037         ],
8038         "type": "string"
8039     },
8040     "minItems": 1,
8041     "readOnly": true,
8042     "type": "array"
8043 }
8044 },
8045 "type" : "object",
8046 "required": ["dos", "isop", "cm", "tm", "om", "sm", "rowneruuid"]
8047 },
8048 "Pstat-Update" : {
8049     "properties": {
8050         "rowneruuid": {
8051             "description": "The UUID formatted identity of the Resource owner\nFormat pattern
8052 according to IETF RFC 4122.",
8053             "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}$",
8054             "type": "string"
8055         },
8056         "om": {
8057             "description": "Current operational mode\nDevice provisioning operation may be server
8058 directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer
8059 and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning
8060 services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8
8061 - Unused16 - Unused32 - Unused64 - Unused128 - Unused",
8062             "maximum": 7,
8063             "minimum": 1,
8064             "type": "integer"
8065         },
8066         "tm": {
8067             "description": "Target device provisioning mode\nDevice provisioning mode maintains a
8068 bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character
8069 in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2
8070 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management
8071 services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate
8072 Software Version Validation128 - Initiate Secure Software Update",
8073             "maximum": 255,
8074             "minimum": 0,
8075             "type": "integer"
8076         },
8077         "dos": {
8078             "description": "Device on-boarding state\nDevice operation state machine",
8079             "properties": {
8080                 "p": {
8081                     "default": true,
8082                     "description": "'p' is TRUE when the 's' state is pending until all necessary changes
8083 to device resources are complete.",
8084                     "readOnly": true,
8085                     "type": "boolean"
8086                 },

```

```

8087     "s": {
8088         "description": "The current or pending operational state",
8089         "x-detail-desc": [
8090             "0 - RESET - Device reset state",
8091             "1 - RFOTM - Ready for device owner transfer method state",
8092             "2 - RFPPO - Ready for device provisioning state",
8093             "3 - RFNOP - Ready for device normal operation state",
8094             "4 - SRESET - The device is in a soft reset state"
8095         ],
8096         "maximum": 4,
8097         "minimum": 0,
8098         "type": "integer"
8099     },
8100 },
8101 "required": [
8102     "s"
8103 ],
8104 "type": "object"
8105 },
8106 },
8107 "type" : "object"
8108 }
8109 }
8110 }
8111 }

```

8112 **C.10.5 Property definition**

8113 Table C.18 defines the Properties that are part of the ['oic.r.pstat'] Resource Type

8114 **Table C.18 – The Property definitions of the Resource with type 'rt' = ['oic.r.pstat']**

Property name	Value type	Mandatory	Access mode	Description
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
rowneruuid	string	Yes	Read Write	The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.
sm	integer	Yes	Read Only	Supported operational modes Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed

				utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client-directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused
rt	array: schema see	No	Read Only	Resource Type of the Resource
isop	boolean	Yes	Read Only	true indicates device is operational
tm	integer	Yes	Read Write	Target device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a device. The value can be either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version

				Validation128 - Initiate Secure Software Update
n	multiple types: see schema	No	Read Write	
dos	object: see schema	Yes	Read Write	Device on- boarding state Device operation state machine
id	multiple types: see schema	No	Read Write	
om	integer	Yes	Read Write	Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client- directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused
cm	integer	Yes	Read Only	Current device provisioning mode Device provisioning mode maintains a bitmask of the possible provisioning states of a device. The value can be

				<p>either 8 or 16 character in length. If its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update</p>
dos	object: see schema	No	Read Write	<p>Device on-boarding state Device operation state machine</p>
rowneruuid	string	No	Read Write	<p>The UUID formatted identity of the Resource owner Format pattern according to IETF RFC 4122.</p>
om	integer	No	Read Write	<p>Current operational mode Device provisioning operation may be server directed or client (aka provisioning service) directed. The value is a bitmask encoded as integer and indicates the provisioning operation</p>

				modes1 - Server-directed utilizing multiple provisioning services2 - Server-directed utilizing a single provisioning service4 - Client- directed provisioning8 - Unused16 - Unused32 - Unused64 - Unused128 - Unused
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 its only 8 characters it represents the lower byte value1 - Manufacturer reset state2 - Device pairing and owner transfer state4 - Unused8 - Provisioning of credential management services16 - Provisioning of access management services32 - Provisioning of local ACLs64 - Initiate Software Version Validation128 - Initiate Secure Software Update

8115 **C.10.6 CRUDN behaviour**

8116 Table C.19 defines the CRUDN operations that are supported on the ['oic.r.pstat'] Resource Type

8117 **Table C.19 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.pstat']**

Create	Read	Update	Delete	Notify
	get	post		observe

8118 **C.11 Asserted Roles**

8119 **C.11.1 Introduction**

8120 This Resource specifies roles that have been asserted.

8121

8122 **C.11.2 Well-known URI**

8123 /oic/sec/roles

8124 **C.11.3 Resource type**

8125 The resource type (rt) is defined as: ['oic.r.roles'].

8126 **C.11.4 OpenAPI 2.0 definition**

```

8127 {
8128   "swagger": "2.0",
8129   "info": {
8130     "title": "Asserted Roles",
8131     "version": "v1.0-20170323",
8132     "license": {
8133       "name": "OCF Data Model License",
8134       "url":
8135         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8136         CENSE.md",
8137       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8138       reserved."
8139     },
8140     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8141   },
8142   "schemes": ["http"],
8143   "consumes": ["application/json"],
8144   "produces": ["application/json"],
8145   "paths": {
8146     "/oic/sec/roles" : {
8147       "get": {
8148         "description": "This Resource specifies roles that have been asserted.\n",
8149         "parameters": [
8150           {"$ref": "#/parameters/interface"}
8151         ],
8152         "responses": {
8153           "200": {
8154             "description": "",
8155             "x-example":
8156               {
8157                 "roles" :[
8158                   {
8159                     "credid":1,
8160                     "credtype":8,
8161                     "subjectuuid":"00000000-0000-0000-0000-000000000000",
8162                     "publicdata":
8163                       {
8164                         "encoding":"oic.sec.encoding.pem",
8165                         "data":"PEMENCODEDROLECERT"
8166                       },
8167                     "optionaldata":
8168                       {
8169                         "revstat": false,
8170                         "encoding":"oic.sec.encoding.pem",
8171                         "data":"PEMENCODEDISSUERCERT"

```

```

8172         }
8173     },
8174     {
8175         "credid":2,
8176         "credtype":8,
8177         "subjectuuid":"00000000-0000-0000-0000-000000000000",
8178         "publicdata":
8179         {
8180             "encoding":"oic.sec.encoding.pem",
8181             "data":"PEMENCODEDROLECERT"
8182         },
8183         "optionaldata":
8184         {
8185             "revstat": false,
8186             "encoding":"oic.sec.encoding.pem",
8187             "data":"PEMENCODEDISSUERCERT"
8188         }
8189     }
8190 ],
8191 "rt":["oic.r.roles"],
8192 "if":["oic.if.baseline"]
8193 }
8194 ,
8195 "schema": { "$ref": "#/definitions/Roles" }
8196 },
8197 "400": {
8198     "description": "The request is invalid."
8199 }
8200 }
8201 },
8202 "post": {
8203     "description": "Update the roles resource, i.e., assert new roles to this server.\n\nNew
8204 role certificates that match an existing certificate (i.e., publicdata\nand optionaldata are the
8205 same) are not added to the resource (and 204 is\nreturned).\n\nThe provided credid values are
8206 ignored, the resource assigns its own.\n",
8207     "parameters": [
8208         { "$ref": "#/parameters/interface" },
8209         {
8210             "name": "body",
8211             "in": "body",
8212             "required": true,
8213             "schema": { "$ref": "#/definitions/Roles-update" },
8214             "x-example":
8215             {
8216                 "roles" :[
8217                     {
8218                         "credid":1,
8219                         "credtype":8,
8220                         "subjectuuid":"00000000-0000-0000-0000-000000000000",
8221                         "publicdata":
8222                         {
8223                             "encoding":"oic.sec.encoding.pem",
8224                             "data":"PEMENCODEDROLECERT"
8225                         },
8226                         "optionaldata":
8227                         {
8228                             "revstat": false,
8229                             "encoding":"oic.sec.encoding.pem",
8230                             "data":"PEMENCODEDISSUERCERT"
8231                         }
8232                     },
8233                     {
8234                         "credid":2,
8235                         "credtype":8,
8236                         "subjectuuid":"00000000-0000-0000-0000-000000000000",
8237                         "publicdata":
8238                         {
8239                             "encoding":"oic.sec.encoding.pem",
8240                             "data":"PEMENCODEDROLECERT"
8241                         },
8242                         "optionaldata":

```

```

8243         {
8244             "revstat": false,
8245             "encoding": "oic.sec.encoding.pem",
8246             "data": "PEMENCODEDISSUERCERT"
8247         }
8248     },
8249     ],
8250     "rt": ["oic.r.roles"],
8251     "if": ["oic.if.baseline"]
8252 }
8253 }
8254 ],
8255 "responses": {
8256     "400": {
8257         "description": "The request is invalid."
8258     },
8259     "204": {
8260         "description": "The roles entry is updated."
8261     }
8262 },
8263 },
8264 "delete": {
8265     "description": "Deletes roles resource entries.\nWhen DELETE is used without query
8266 parameters, all the roles entries are deleted.\nWhen DELETE is used with a query parameter, only the
8267 entries matching\nthe query parameter are deleted.\n",
8268     "parameters": [
8269         {"$ref": "#/parameters/interface"},
8270         {"$ref": "#/parameters/roles-filtered"}
8271     ],
8272     "responses": {
8273         "200": {
8274             "description": "The specified or all roles resource entries have been successfully
8275 deleted."
8276         },
8277         "400": {
8278             "description": "The request is invalid."
8279         }
8280     }
8281 },
8282 },
8283 },
8284 "parameters": {
8285     "interface": {
8286         "in": "query",
8287         "name": "if",
8288         "type": "string",
8289         "enum": ["oic.if.baseline"]
8290     },
8291     "roles-filtered": {
8292         "in": "query",
8293         "name": "credid",
8294         "required": false,
8295         "type": "integer",
8296         "description": "Only applies to the credential with the specified credid",
8297         "x-example": 2112
8298     }
8299 },
8300 "definitions": {
8301     "Roles": {
8302         "properties": {
8303             "rt": {
8304                 "description": "Resource Type of the Resource",
8305                 "items": {
8306                     "maxLength": 64,
8307                     "type": "string",
8308                     "enum": ["oic.r.roles"]
8309                 },
8310                 "minItems": 1,
8311                 "readOnly": true,
8312                 "type": "array"
8313             },

```

```

8314         "n": {
8315             "$ref":
8316 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8317 schema.json#/definitions/n"
8318         },
8319         "id": {
8320             "$ref":
8321 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
8322 schema.json#/definitions/id"
8323         },
8324         "roles": {
8325             "description": "List of role certificates",
8326             "items": {
8327                 "properties": {
8328                     "credid": {
8329                         "description": "Local reference to a credential resource",
8330                         "type": "integer"
8331                     },
8332                     "credtype": {
8333                         "description": "Representation of this credential's type\nCredential Types - Cred
8334 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8335 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificatel6 - PIN or
8336 password32 - Asymmetric encryption key",
8337                         "maximum": 63,
8338                         "minimum": 0,
8339                         "type": "integer"
8340                     },
8341                     "credusage": {
8342                         "description": "A string that provides hints about how/where the cred is used\nThe
8343 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8344 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8345 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
8346                         "enum": [
8347                             "oic.sec.cred.trustca",
8348                             "oic.sec.cred.cert",
8349                             "oic.sec.cred.rolecert",
8350                             "oic.sec.cred.mfgtrustca",
8351                             "oic.sec.cred.mfgcert"
8352                         ],
8353                         "type": "string"
8354                     },
8355                     "crms": {
8356                         "description": "The refresh methods that may be used to update this credential",
8357                         "items": {
8358                             "description": "Each enum represents a method by which the credentials are
8359 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8360 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8361 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8362 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
8363                             "enum": [
8364                                 "oic.sec.crm.pro",
8365                                 "oic.sec.crm.psk",
8366                                 "oic.sec.crm.rdp",
8367                                 "oic.sec.crm.skdc",
8368                                 "oic.sec.crm.pk10"
8369                             ],
8370                             "type": "string"
8371                         },
8372                         "type": "array"
8373                     },
8374                     "optionaldata": {
8375                         "description": "Credential revocation status information\nOptional credential
8376 contents describes revocation status for this credential",
8377                         "properties": {
8378                             "data": {
8379                                 "description": "The encoded structure",
8380                                 "type": "string"
8381                             },
8382                             "encoding": {
8383                                 "description": "A string specifying the encoding format of the data contained in
8384 the optdata",

```

```

8385         "x-detail-desc": [
8386             "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8387             "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8388             "oic.sec.encoding.base64 - Base64 encoded object",
8389             "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8390             "oic.sec.encoding.der - Encoding for DER encoded certificate",
8391             "oic.sec.encoding.raw - Raw hex encoded data"
8392         ],
8393         "enum": [
8394             "oic.sec.encoding.jwt",
8395             "oic.sec.encoding.cwt",
8396             "oic.sec.encoding.base64",
8397             "oic.sec.encoding.pem",
8398             "oic.sec.encoding.der",
8399             "oic.sec.encoding.raw"
8400         ],
8401         "type": "string"
8402     },
8403     "revstat": {
8404         "description": "Revocation status flag - true = revoked",
8405         "type": "boolean"
8406     }
8407 },
8408 "required": [
8409     "revstat"
8410 ],
8411 "type": "object"
8412 },
8413 "period": {
8414     "description": "String with IETF RFC 5545 Period",
8415     "type": "string"
8416 },
8417 "privatedata": {
8418     "description": "Private credential information\nCredential resource non-public
8419 contents",
8420     "properties": {
8421         "data": {
8422             "description": "The encoded value",
8423             "maxLength": 3072,
8424             "type": "string"
8425         },
8426         "encoding": {
8427             "description": "A string specifying the encoding format of the data contained in
8428 the privdata",
8429             "x-detail-desc": [
8430                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8431                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8432                 "oic.sec.encoding.base64 - Base64 encoded object",
8433                 "oic.sec.encoding.uri - URI reference",
8434                 "oic.sec.encoding.handle - Data is contained in a storage sub-system
8435 referenced using a handle",
8436                 "oic.sec.encoding.raw - Raw hex encoded data"
8437             ],
8438             "enum": [
8439                 "oic.sec.encoding.jwt",
8440                 "oic.sec.encoding.cwt",
8441                 "oic.sec.encoding.base64",
8442                 "oic.sec.encoding.uri",
8443                 "oic.sec.encoding.handle",
8444                 "oic.sec.encoding.raw"
8445             ],
8446             "type": "string"
8447         },
8448         "handle": {
8449             "description": "Handle to a key storage resource",
8450             "type": "integer"
8451         }
8452     },
8453     "required": [
8454         "encoding"
8455     ],

```

```

8456         "type": "object"
8457     },
8458     "publicdata": {
8459         "description": "Public credential information",
8460         "properties": {
8461             "data": {
8462                 "description": "The encoded value",
8463                 "maxLength": 3072,
8464                 "type": "string"
8465             },
8466             "encoding": {
8467                 "description": "A string specifying the encoding format of the data contained in
8468 the pubdata",
8469                 "x-detail-desc": [
8470                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8471                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8472                     "oic.sec.encoding.base64 - Base64 encoded object",
8473                     "oic.sec.encoding.uri - URI reference",
8474                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8475                     "oic.sec.encoding.der - Encoding for DER encoded certificate",
8476                     "oic.sec.encoding.raw - Raw hex encoded data"
8477                 ],
8478                 "enum": [
8479                     "oic.sec.encoding.jwt",
8480                     "oic.sec.encoding.cwt",
8481                     "oic.sec.encoding.base64",
8482                     "oic.sec.encoding.uri",
8483                     "oic.sec.encoding.pem",
8484                     "oic.sec.encoding.der",
8485                     "oic.sec.encoding.raw"
8486                 ],
8487                 "type": "string"
8488             }
8489         },
8490         "type": "object"
8491     },
8492     "roleid": {
8493         "description": "The role this credential possesses\nSecurity role specified as an
8494 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
8495         "properties": {
8496             "authority": {
8497                 "description": "The Authority component of the entity being identified. A NULL
8498 <Authority> refers to the local entity or device.",
8499                 "type": "string"
8500             },
8501             "role": {
8502                 "description": "The ID of the role being identified.",
8503                 "type": "string"
8504             }
8505         },
8506         "required": [
8507             "role"
8508         ],
8509         "type": "object"
8510     },
8511     "subjectuuid": {
8512         "anyOf": [
8513             {
8514                 "description": "The id of the device, which the cred entry applies to or \"*\n
8515 for wildcard identity",
8516                 "pattern": "^\\*$",
8517                 "type": "string"
8518             },
8519             {
8520                 "description": "Format pattern according to IETF RFC 4122.",
8521                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8522 F0-9]{12}$",
8523                 "type": "string"
8524             }
8525         ]
8526     }

```

```

8527         },
8528         "type": "object"
8529     },
8530     "type": "array"
8531 },
8532 "if": {
8533     "description": "The interface set supported by this resource",
8534     "items": {
8535         "enum": [
8536             "oic.if.baseline"
8537         ],
8538         "type": "string"
8539     },
8540     "minItems": 1,
8541     "readOnly": true,
8542     "type": "array"
8543 }
8544 },
8545 "type": "object",
8546 "required": ["roles"]
8547 },
8548 "Roles-update" : {
8549     "properties": {
8550         "roles": {
8551             "description": "List of role certificates",
8552             "items": {
8553                 "properties": {
8554                     "credid": {
8555                         "description": "Local reference to a credential resource",
8556                         "type": "integer"
8557                     },
8558                     "credtype": {
8559                         "description": "Representation of this credential's type\nCredential Types - Cred
8560 type encoded as a bitmask.0 - Empty credential used for testing1 - Symmetric pair-wise key2 -
8561 Symmetric group key4 - Asymmetric signing key8 - Asymmetric signing key with certificate16 - PIN or
8562 password32 - Asymmetric encryption key",
8563                         "maximum": 63,
8564                         "minimum": 0,
8565                         "type": "integer"
8566                     },
8567                     "credusage": {
8568                         "description": "A string that provides hints about how/where the cred is used\nThe
8569 type of credusage.oic.sec.cred.trustca - Trust certificateoic.sec.cred.cert -
8570 Certificateoic.sec.cred.rolecert - Role Certificateoic.sec.cred.mfgtrustca - Manufacturer
8571 Certificate Trust Anchoroic.sec.cred.mfgcert - Manufacturer Certificate",
8572                         "enum": [
8573                             "oic.sec.cred.trustca",
8574                             "oic.sec.cred.cert",
8575                             "oic.sec.cred.rolecert",
8576                             "oic.sec.cred.mfgtrustca",
8577                             "oic.sec.cred.mfgcert"
8578                         ],
8579                         "type": "string"
8580                     },
8581                     "crms": {
8582                         "description": "The refresh methods that may be used to update this credential",
8583                         "items": {
8584                             "description": "Each enum represents a method by which the credentials are
8585 refreshed.oic.sec.crm.pro - Credentials refreshed by a provisioning serviceoic.sec.crm.rdp -
8586 Credentials refreshed by a key agreement protocol and random PINoic.sec.crm.psk - Credentials
8587 refreshed by a key agreement protocoloic.sec.crm.skdc - Credentials refreshed by a key distribution
8588 serviceoic.sec.crm.pk10 - Credentials refreshed by a PKCS#10 request to a CA",
8589                             "enum": [
8590                                 "oic.sec.crm.pro",
8591                                 "oic.sec.crm.psk",
8592                                 "oic.sec.crm.rdp",
8593                                 "oic.sec.crm.skdc",
8594                                 "oic.sec.crm.pk10"
8595                             ],
8596                             "type": "string"
8597                         }
8598                     }
8599                 }

```

```

8598         "type": "array"
8599     },
8600     "optionaldata": {
8601         "description": "Credential revocation status information\nOptional credential
8602 contents describes revocation status for this credential",
8603         "properties": {
8604             "data": {
8605                 "description": "The encoded structure",
8606                 "type": "string"
8607             },
8608             "encoding": {
8609                 "description": "A string specifying the encoding format of the data contained in
8610 the optdata",
8611                 "x-detail-desc": [
8612                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8613                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8614                     "oic.sec.encoding.base64 - Base64 encoded object",
8615                     "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8616                     "oic.sec.encoding.der - Encoding for DER encoded certificate",
8617                     "oic.sec.encoding.raw - Raw hex encoded data"
8618                 ],
8619                 "enum": [
8620                     "oic.sec.encoding.jwt",
8621                     "oic.sec.encoding.cwt",
8622                     "oic.sec.encoding.base64",
8623                     "oic.sec.encoding.pem",
8624                     "oic.sec.encoding.der",
8625                     "oic.sec.encoding.raw"
8626                 ],
8627                 "type": "string"
8628             },
8629             "revstat": {
8630                 "description": "Revocation status flag - true = revoked",
8631                 "type": "boolean"
8632             }
8633         },
8634         "required": [
8635             "revstat"
8636         ],
8637         "type": "object"
8638     },
8639     "period": {
8640         "description": "String with IETF RFC 5545 Period",
8641         "type": "string"
8642     },
8643     "privatedata": {
8644         "description": "Private credential information\nCredential resource non-public
8645 contents",
8646         "properties": {
8647             "data": {
8648                 "description": "The encoded value",
8649                 "maxLength": 3072,
8650                 "type": "string"
8651             },
8652             "encoding": {
8653                 "description": "A string specifying the encoding format of the data contained in
8654 the privdata",
8655                 "x-detail-desc": [
8656                     "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8657                     "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8658                     "oic.sec.encoding.base64 - Base64 encoded object",
8659                     "oic.sec.encoding.uri - URI reference",
8660                     "oic.sec.encoding.handle - Data is contained in a storage sub-system
8661 referenced using a handle",
8662                     "oic.sec.encoding.raw - Raw hex encoded data"
8663                 ],
8664                 "enum": [
8665                     "oic.sec.encoding.jwt",
8666                     "oic.sec.encoding.cwt",
8667                     "oic.sec.encoding.base64",
8668                     "oic.sec.encoding.uri"

```

```

8669         "oic.sec.encoding.handle",
8670         "oic.sec.encoding.raw"
8671     ],
8672     "type": "string"
8673 },
8674 "handle": {
8675     "description": "Handle to a key storage resource",
8676     "type": "integer"
8677 }
8678 },
8679 "required": [
8680     "encoding"
8681 ],
8682 "type": "object"
8683 },
8684 "publicdata": {
8685     "description": "Public credential information",
8686     "properties": {
8687         "data": {
8688             "description": "The encoded value",
8689             "maxLength": 3072,
8690             "type": "string"
8691         },
8692         "encoding": {
8693             "description": "A string specifying the encoding format of the data contained in
8694 the pubdata",
8695             "x-detail-desc": [
8696                 "oic.sec.encoding.jwt - IETF RFC 7519 JSON web token (JWT) encoding",
8697                 "oic.sec.encoding.cwt - IETF RFC 8392 CBOR web token (CWT) encoding",
8698                 "oic.sec.encoding.base64 - Base64 encoded object",
8699                 "oic.sec.encoding.uri - URI reference",
8700                 "oic.sec.encoding.pem - Encoding for PEM encoded certificate or chain",
8701                 "oic.sec.encoding.der - Encoding for DER encoded certificate",
8702                 "oic.sec.encoding.raw - Raw hex encoded data"
8703             ],
8704             "enum": [
8705                 "oic.sec.encoding.jwt",
8706                 "oic.sec.encoding.cwt",
8707                 "oic.sec.encoding.base64",
8708                 "oic.sec.encoding.uri",
8709                 "oic.sec.encoding.pem",
8710                 "oic.sec.encoding.der",
8711                 "oic.sec.encoding.raw"
8712             ],
8713             "type": "string"
8714         }
8715     },
8716     "type": "object"
8717 },
8718 "roleid": {
8719     "description": "The role this credential possesses\nSecurity role specified as an
8720 <Authority> & <Rolename>. A NULL <Authority> refers to the local entity or device.",
8721     "properties": {
8722         "authority": {
8723             "description": "The Authority component of the entity being identified. A NULL
8724 <Authority> refers to the local entity or device.",
8725             "type": "string"
8726         },
8727         "role": {
8728             "description": "The ID of the role being identified.",
8729             "type": "string"
8730         }
8731     },
8732     "required": [
8733         "role"
8734 ],
8735     "type": "object"
8736 },
8737 "subjectuuid": {
8738     "anyOf": [
8739         {

```

```

8740         "description": "The id of the device, which the cred entry applies to or \"*\
8741 for wildcard identity",
8742         "pattern": "^\\*$",
8743         "type": "string"
8744     },
8745     {
8746         "description": "Format pattern according to IETF RFC 4122.",
8747         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-
8748 F0-9]{12}$",
8749         "type": "string"
8750     }
8751 ]
8752 },
8753 },
8754 "type": "object"
8755 },
8756 "type": "array"
8757 }
8758 },
8759 "type": "object",
8760 "required": ["roles"]
8761 }
8762 }
8763 }
8764

```

8765 **C.11.5 Property definition**

8766 Table C.20 defines the Properties that are part of the ['oic.r.roles'] Resource Type

8767 **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	
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
roles	array: see schema	Yes	Read Write	List of role certificates
id	multiple types: see schema	No	Read Write	

8768 **C.11.6 CRUDN behaviour**

8769 Table C.21 defines the CRUDN operations that are supported on the ['oic.r.roles'] Resource Type

8770 **Table C.21 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.roles']**

Create	Read	Update	Delete	Notify
	get	post	delete	observe

8771 **C.12 Signed Access Control List**

8772 **C.12.1 Introduction**

8773 This Resource specifies a signed ACL object.

8774

8775 **C.12.2 Well-known URI**

8776 /oic/sec/sacl

8777 **C.12.3 Resource type**

8778 The resource type (rt) is defined as: ['oic.r.sacl'].

8779 **C.12.4 OpenAPI 2.0 definition**

```

8780 {
8781   "swagger": "2.0",
8782   "info": {
8783     "title": "Signed Access Control List",
8784     "version": "v1.0-20150819",
8785     "license": {
8786       "name": "OCF Data Model License",
8787       "url":
8788         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
8789         CENSE.md",
8790       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
8791         reserved."
8792     },
8793     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
8794   },
8795   "schemes": ["http"],
8796   "consumes": ["application/json"],
8797   "produces": ["application/json"],
8798   "paths": {
8799     "/oic/sec/sacl" : {
8800       "get": {
8801         "description": "This Resource specifies a signed ACL object.\n",
8802         "parameters": [
8803           {"$ref": "#/parameters/interface"}
8804         ],
8805         "responses": {
8806           "200": {
8807             "description": "",
8808             "x-example":
8809               {
8810                 "rt": ["oic.r.sacl"],
8811                 "aclist2": [
8812                   {
8813                     "aceid": 1,
8814                     "subject": {"uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9"},
8815                     "resources": [
8816                       {
8817                         "href": "/temp",
8818                         "rt": ["oic.r.temperature"],
8819                         "if": ["oic.if.baseline", "oic.if.a"]
8820                       },
8821                       {
8822                         "href": "/temp",
8823                         "rt": ["oic.r.temperature"],
8824                         "if": ["oic.if.baseline", "oic.if.s"]
8825                       }
8826                     ],
8827                     "permission": 31,
8828                     "validity": [
8829                       {
8830                         "period": "20160101T180000Z/20170102T070000Z",
8831                         "recurrence": [ "DSTART:XXXXX",
8832 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8833                       },
8834                       {
8835                         "period": "20160101T180000Z/PT5H30M",
8836                         "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8837                       }
8838                     ]
8839                   }
8840                 ],
8841                 "aceid": 2,

```

```

8842         "subject": {
8843             "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8844             "role": "SOME_STRING"
8845         },
8846         "resources": [
8847             {
8848                 "href": "/light",
8849                 "rt": ["oic.r.light"],
8850                 "if": ["oic.if.baseline", "oic.if.a"]
8851             },
8852             {
8853                 "href": "/door",
8854                 "rt": ["oic.r.door"],
8855                 "if": ["oic.if.baseline", "oic.if.a"]
8856             }
8857         ],
8858         "permission": 15
8859     }
8860 ],
8861     "signature": {
8862         "sigtype": "oic.sec.sigtype.pk7",
8863         "sigvalue": "ENCODED-SIGNATURE-VALUE"
8864     }
8865 },
8866 "schema": { "$ref": "#/definitions/Sacl" }
8867 }
8868 },
8869 },
8870 "post": {
8871     "description": "Sets the sacl resource data\n",
8872     "parameters": [
8873         { "$ref": "#/parameters/interface" },
8874         {
8875             "name": "body",
8876             "in": "body",
8877             "required": true,
8878             "schema": { "$ref": "#/definitions/Sacl" },
8879             "x-example":
8880             {
8881                 "aclist2": [
8882                     {
8883                         "aceid": 1,
8884                         "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8885                         "resources": [
8886                             {
8887                                 "href": "/temp",
8888                                 "rt": ["oic.r.temperature"],
8889                                 "if": ["oic.if.baseline", "oic.if.a"]
8890                             },
8891                             {
8892                                 "href": "/temp",
8893                                 "rt": ["oic.r.temperature"],
8894                                 "if": ["oic.if.baseline", "oic.if.s"]
8895                             }
8896                         ],
8897                         "permission": 31,
8898                         "validity": [
8899                             {
8900                                 "period": "20160101T180000Z/20170102T070000Z",
8901                                 "recurrence": [ "DSTART:XXXXX",
8902 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8903                             },
8904                             {
8905                                 "period": "20160101T180000Z/PT5H30M",
8906                                 "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8907                             }
8908                         ]
8909                     }
8910                 ],
8911                 "aceid": 2,
8912                 "subject": {

```

```

8913         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8914         "role": "SOME_STRING"
8915     },
8916     "resources": [
8917         {
8918             "href": "/light",
8919             "rt": ["oic.r.light"],
8920             "if": ["oic.if.baseline", "oic.if.a"]
8921         },
8922         {
8923             "href": "/door",
8924             "rt": ["oic.r.door"],
8925             "if": ["oic.if.baseline", "oic.if.a"]
8926         }
8927     ],
8928     "permission": 15
8929 }
8930 ],
8931 "signature": {
8932     "sigtype": "oic.sec.sigtype.pk7",
8933     "sigvalue": "ENCODED-SIGNATURE-VALUE"
8934 }
8935 }
8936 },
8937 ],
8938 "responses": {
8939     "400": {
8940         "description": "The request is invalid."
8941     },
8942     "201": {
8943         "description": "The ACL entry is created."
8944     },
8945     "204": {
8946         "description": "The ACL entry is updated."
8947     }
8948 },
8949 },
8950 "put": {
8951     "description": "Sets the sacl resource data\n",
8952     "parameters": [
8953         {"$ref": "#/parameters/interface"},
8954         {
8955             "name": "body",
8956             "in": "body",
8957             "required": true,
8958             "schema": { "$ref": "#/definitions/Sacl" },
8959             "x-example":
8960             {
8961                 "acllist2": [
8962                     {
8963                         "aceid": 1,
8964                         "subject": { "uuid": "e61c3e6b-9c54-4b81-8ce5-f9039c1d04d9" },
8965                         "resources": [
8966                             {
8967                                 "href": "/temp",
8968                                 "rt": ["oic.r.temperature"],
8969                                 "if": ["oic.if.baseline", "oic.if.a"]
8970                             },
8971                             {
8972                                 "href": "/temp",
8973                                 "rt": ["oic.r.temperature"],
8974                                 "if": ["oic.if.baseline", "oic.if.s"]
8975                             }
8976                         ],
8977                         "permission": 31,
8978                         "validity": [
8979                             {
8980                                 "period": "20160101T180000Z/20170102T070000Z",
8981                                 "recurrence": [ "DSTART:XXXXX",
8982 "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8983                             }
8984                         ]
8985                     }
8986                 ]
8987             }
8988         }
8989     ]
8990 }

```

```

8984         {
8985             "period": "20160101T180000Z/PT5H30M",
8986             "recurrence": [ "RRULE:FREQ=DAILY;UNTIL=20180131T140000Z;BYMONTH=1" ]
8987         }
8988     ]
8989 },
8990 {
8991     "aceid": 2,
8992     "subject": {
8993         "authority": "484b8a51-cb23-46c0-a5f1-b4aebef50ebe",
8994         "role": "SOME_STRING"
8995     },
8996     "resources": [
8997         {
8998             "href": "/light",
8999             "rt": ["oic.r.light"],
9000             "if": ["oic.if.baseline", "oic.if.a"]
9001         },
9002         {
9003             "href": "/door",
9004             "rt": ["oic.r.door"],
9005             "if": ["oic.if.baseline", "oic.if.a"]
9006         }
9007     ],
9008     "permission": 15
9009 }
9010 ],
9011 "signature": {
9012     "sigtype": "oic.sec.sigtype.pk7",
9013     "sigvalue": "ENCODED-SIGNATURE-VALUE"
9014 }
9015 }
9016 },
9017 ],
9018 "responses": {
9019     "400": {
9020         "description": "The request is invalid."
9021     },
9022     "201": {
9023         "description": "The signed ACL entry is created."
9024     }
9025 },
9026 },
9027 "delete": {
9028     "description": "Deletes the signed ACL data.\nWhen DELETE is used without query parameters,
9029 the entire collection is deleted.\nWhen DELETE is used with the query parameter where \"acl\" is
9030 specified, only the matched entry is deleted.\n",
9031     "parameters": [
9032         { "$ref": "#/parameters/interface" },
9033         {
9034             "in": "query",
9035             "description": "Delete the signed ACL identified by the string containing subject
9036 UUID.\n",
9037             "type": "string",
9038             "name": "subject"
9039         }
9040     ],
9041     "responses": {
9042         "200": {
9043             "description": "The signed ACL instance or the the entire signed ACL resource has
9044 been successfully deleted."
9045         },
9046         "400": {
9047             "description": "The request is invalid."
9048         }
9049     }
9050 }
9051 },
9052 },
9053 "parameters": {
9054     "interface": {

```

```

9055     "in" : "query",
9056     "name" : "if",
9057     "type" : "string",
9058     "enum" : ["oic.if.baseline"]
9059   }
9060 },
9061 "definitions": {
9062   "Sacl" : {
9063     "properties": {
9064       "rt": {
9065         "description": "Resource Type of the Resource",
9066         "items": {
9067           "maxLength": 64,
9068           "type": "string",
9069           "enum": ["oic.r.sacl"]
9070         },
9071         "minItems": 1,
9072         "readOnly": true,
9073         "type": "array"
9074       },
9075       "aclist2": {
9076         "description": "Access Control Entries in the Acl resource",
9077         "items": {
9078           "properties": {
9079             "aceid": {
9080               "description": "An identifier for the ACE that is unique within the ACL. In cases
9081 where it isn't supplied in an update, the Server will add the ACE and assign it a unique value.",
9082               "minimum": 1,
9083               "type": "integer"
9084             },
9085             "permission": {
9086               "description": "Bitmask encoding of CRUDN permission\nThe encoded bitmask indicating
9087 permissions",
9088               "x-detail-desc": [
9089                 "0 - No permissions",
9090                 "1 - Create permission is granted",
9091                 "2 - Read, observe, discover permission is granted",
9092                 "4 - Write, update permission is granted",
9093                 "8 - Delete permission is granted",
9094                 "16 - Notify permission is granted"
9095               ],
9096               "maximum": 31,
9097               "minimum": 0,
9098               "type": "integer"
9099             },
9100             "resources": {
9101               "description": "References the application's resources to which a security policy
9102 applies",
9103               "items": {
9104                 "description": "Each resource must have at least one of these properties set",
9105                 "properties": {
9106                   "href": {
9107                     "allOf": [
9108                       {
9109                         "description": "When present, the ACE only applies when the href matches"
9110                       },
9111                       {
9112                         "description": "This is the target URI, it can be specified as a Relative
9113 Reference or fully-qualified URI.",
9114                         "format": "uri",
9115                         "maxLength": 256,
9116                         "type": "string"
9117                       }
9118                     ]
9119                   },
9120                   "if": {
9121                     "description": "When present, the ACE only applies when the if (interface)
9122 matches\nThe interface set supported by this resource",
9123                     "items": {
9124                       "enum": [
9125                         "oic.if.baseline",

```

```

9126         "oic.if.ll",
9127         "oic.if.b",
9128         "oic.if.rw",
9129         "oic.if.r",
9130         "oic.if.a",
9131         "oic.if.s"
9132     ],
9133     "type": "string"
9134 },
9135     "minItems": 1,
9136     "type": "array"
9137 },
9138     "rt": {
9139         "description": "When present, the ACE only applies when the rt (resource type)
9140 matches\nResource Type of the Resource",
9141         "items": {
9142             "maxLength": 64,
9143             "type": "string"
9144         },
9145         "minItems": 1,
9146         "type": "array"
9147     },
9148     "wc": {
9149         "description": "A wildcard matching policy",
9150         "pattern": "^[+*]$",
9151         "type": "string"
9152     }
9153 },
9154     "type": "object"
9155 },
9156     "type": "array"
9157 },
9158     "subject": {
9159         "anyOf": [
9160             {
9161                 "description": "Device identifier",
9162                 "properties": {
9163                     "uuid": {
9164                         "description": "A UUID Device ID\nFormat pattern according to IETF RFC
9165 4122.",
9166                         "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
9167 fA-F0-9]{12}$",
9168                         "type": "string"
9169                     }
9170                 },
9171                 "required": [
9172                     "uuid"
9173                 ],
9174                 "type": "object"
9175             },
9176             {
9177                 "description": "Security role specified as an <Authority> & <Rolename>. A NULL
9178 <Authority> refers to the local entity or device.",
9179                 "properties": {
9180                     "authority": {
9181                         "description": "The Authority component of the entity being identified. A
9182 NULL <Authority> refers to the local entity or device.",
9183                         "type": "string"
9184                     },
9185                     "role": {
9186                         "description": "The ID of the role being identified.",
9187                         "type": "string"
9188                     }
9189                 },
9190                 "required": [
9191                     "role"
9192                 ],
9193                 "type": "object"
9194             },
9195             {
9196                 "properties": {

```

```

9197         "conntype": {
9198             "description": "This property allows an ACE to be matched based on the
9199 connection or message type",
9200             "x-detail-desc": [
9201                 "auth-crypt - ACE applies if the Client is authenticated and the data
9202 channel or message is encrypted and integrity protected",
9203                 "anon-clear - ACE applies if the Client is not authenticated and the data
9204 channel or message is not encrypted but may be integrity protected"
9205             ],
9206             "enum": [
9207                 "auth-crypt",
9208                 "anon-clear"
9209             ],
9210             "type": "string"
9211         }
9212     },
9213     "required": [
9214         "conntype"
9215     ],
9216     "type": "object"
9217 }
9218 ]
9219 },
9220 "validity": {
9221     "description": "validity is an array of time-pattern objects",
9222     "items": {
9223         "description": "The time-pattern contains a period and recurrence expressed in
9224 IETF RFC 5545 syntax",
9225         "properties": {
9226             "period": {
9227                 "description": "String represents a period using the IETF RFC 5545 Period",
9228                 "type": "string"
9229             },
9230             "recurrence": {
9231                 "description": "String array represents a recurrence rule using the IETF RFC
9232 5545 Recurrence",
9233                 "items": {
9234                     "type": "string"
9235                 },
9236                 "type": "array"
9237             }
9238         },
9239         "required": [
9240             "period"
9241         ],
9242         "type": "object"
9243     },
9244     "type": "array"
9245 }
9246 },
9247 "required": [
9248     "aceid",
9249     "resources",
9250     "permission",
9251     "subject"
9252 ],
9253 "type": "object"
9254 },
9255 "type": "array"
9256 },
9257 "n": {
9258     "$ref":
9259 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9260 schema.json#/definitions/n"
9261 },
9262 "id": {
9263     "$ref":
9264 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9265 schema.json#/definitions/id"
9266 },
9267 "signature": {

```

```

9268     "description": "The signature over the ACL resource\nEncoded signature data",
9269     "properties": {
9270         "sigtype": {
9271             "description": "The string specifies the predefined signature format",
9272             "x-detail-desc": [
9273                 "IETF RFC 7515 JSON web signature (JWS) object",
9274                 "IETF RFC 2315 base64 encoded object",
9275                 "CBOR encoded JWS object"
9276             ],
9277             "enum": [
9278                 "oic.sec.sigtype.jws",
9279                 "oic.sec.sigtype.pk7",
9280                 "oic.sec.sigtype.cws"
9281             ],
9282             "type": "string"
9283         },
9284         "sigvalue": {
9285             "description": "The encoded signature",
9286             "type": "string"
9287         }
9288     },
9289     "required": [
9290         "sigtype",
9291         "sigvalue"
9292     ],
9293     "type": "object"
9294 },
9295 "if": {
9296     "description": "The interface set supported by this resource",
9297     "items": {
9298         "enum": [
9299             "oic.if.baseline"
9300         ],
9301         "type": "string"
9302     },
9303     "minItems": 1,
9304     "readOnly": true,
9305     "type": "array"
9306 },
9307 },
9308 "type" : "object",
9309 "required": ["aclist2", "signature"]
9310 }
9311 }
9312 }
9313

```

9314 **C.12.5 Property definition**

9315 Table C.22 defines the Properties that are part of the ['oic.r.sacl'] Resource Type

9316 **Table C.22 – The Property definitions of the Resource with type 'rt' = ['oic.r.sacl']**

Property name	Value type	Mandatory	Access mode	Description
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
aclist2	array: see schema	Yes	Read Write	Access Control Entries in the Acl resource
signature	object: see schema	Yes	Read Write	The signature over the ACL resource Encoded signature data
n	multiple types:	No	Read Write	

	see schema			
if	array: schema	see	No	Read Only
				The OCF Interface set supported by this resource

9317 **C.12.6 CRUDN behaviour**

9318 Table C.23 defines the CRUDN operations that are supported on the ['oic.r.sacl'] Resource Type

9319 **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

9320 **C.13 Session**

9321 **C.13.1 Introduction**

9322 Resource that manages the persistent session between a Device and OCF Cloud

9323 **C.13.2 Well-known URI**

9324 /oic/sec/session

9325 **C.13.3 Resource type**

9326 The resource type (rt) is defined as: ['oic.r.session'].

9327 **C.13.4 OpenAPI 2.0 definition**

```

9328 {
9329   "swagger": "2.0",
9330   "info": {
9331     "title": "Session",
9332     "version": "v1.0-20181001",
9333     "license": {
9334       "name": "OCF Data Model License",
9335       "url":
9336         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9337         CENSE.md",
9338       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9339         reserved."
9340     },
9341     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9342   },
9343   "schemes": ["http"],
9344   "consumes": ["application/json"],
9345   "produces": ["application/json"],
9346   "paths": {
9347     "/oic/sec/session" : {
9348       "post": {
9349         "description": "Resource that manages the persistent session between a Device and OCF Cloud",
9350         "parameters": [
9351           {"$ref": "#/parameters/interface"},
9352           {
9353             "name": "body",
9354             "in": "body",
9355             "required": true,
9356             "schema": { "$ref": "#/definitions/Account-Session-Request" },
9357             "x-example":
9358               {
9359                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9360                 "di" : "9cfbeb8e-5ale-4d1c-9d01-00c04fd430c8",
9361                 "accesstoken" : "0f3d9f7fe5491d54077d",
9362                 "login" : true
9363               }
9364           }
9365         ],
9366         "responses": {

```

```

9367         "204": {
9368             "description": "",
9369             "x-example":
9370                 {
9371                     "rt": ["oic.r.session"],
9372                     "expiresin": 3600
9373                 },
9374             "schema": { "$ref": "#/definitions/Account-Session-Response" }
9375         }
9376     }
9377 }
9378 },
9379 },
9380 "parameters": {
9381     "interface": {
9382         "in": "query",
9383         "name": "if",
9384         "type": "string",
9385         "enum": ["oic.if.baseline"]
9386     }
9387 },
9388 "definitions": {
9389     "Account-Session-Request": {
9390         "properties": {
9391             "uid": {
9392                 "description": "Format pattern according to IETF RFC 4122.",
9393                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9394                 "type": "string"
9395             },
9396             "di": {
9397                 "description": "The device ID\nFormat pattern according to IETF RFC 4122.",
9398                 "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}$",
9399                 "type": "string"
9400             },
9401             "accesstoken": {
9402                 "description": "Access-Token used to grant access right for the device to sign-in",
9403                 "pattern": "(?!$|\\s+).*",
9404                 "type": "string"
9405             },
9406             "login": {
9407                 "description": "Action for the request: true = login, false = logout",
9408                 "type": "boolean"
9409             }
9410         },
9411         "type": "object",
9412         "required": ["uid", "di", "accesstoken", "login"]
9413     },
9414     "Account-Session-Response": {
9415         "properties": {
9416             "expiresin": {
9417                 "description": "Access-Token remaining life time in seconds (-1 if permanent)",
9418                 "readOnly": true,
9419                 "type": "integer"
9420             },
9421             "rt": {
9422                 "description": "Resource Type of the Resource",
9423                 "items": {
9424                     "maxLength": 64,
9425                     "type": "string",
9426                     "enum": ["oic.r.session"]
9427                 },
9428                 "minItems": 1,
9429                 "readOnly": true,
9430                 "type": "array"
9431             },
9432             "n": {
9433                 "$ref":
9434                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9435                 schema.json#/definitions/n"
9436             },
9437             "id": {

```

```

9438     "$ref":
9439     "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9440     schema.json#/definitions/id"
9441   },
9442   "if": {
9443     "description": "The interface set supported by this resource",
9444     "items": {
9445       "enum": [
9446         "oic.if.baseline"
9447       ],
9448       "type": "string"
9449     },
9450     "minItems": 1,
9451     "readOnly": true,
9452     "type": "array"
9453   }
9454 },
9455 "type" : "object",
9456 "required" : ["expiresin"]
9457 }
9458 }
9459 }
9460

```

9461 C.13.5 Property definition

9462 Table C.24 defines the Properties that are part of the ['oic.r.session'] Resource Type

9463 **Table C.24 – The Property definitions of the Resource with type 'rt' = ['oic.r.session']**

Property name	Value type	Mandatory	Access mode	Description
di	string	Yes	Read Write	The device ID Format pattern according to IETF RFC 4122.
uid	string	Yes	Read Write	Format pattern according to IETF RFC 4122.
login	boolean	Yes	Read Write	Action for the request: true = login, false = logout
accesstoken	string	Yes	Read Write	Access-Token used to grant access right for the device to sign-in
expiresin	integer	Yes	Read Only	Access-Token remaining life time in seconds (-1 if permanent)
n	multiple types: see schema	No	Read Write	
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource

9464 **C.13.6 CRUDN behaviour**

9465 Table C.25 defines the CRUDN operations that are supported on the ['oic.r.session'] Resource
 9466 Type

9467 **Table C.25 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.session']**

Create	Read	Update	Delete	Notify
		post		

9468 **C.14 Security Profile**

9469 **C.14.1 Introduction**

9470 Resource specifying supported and active security profile(s)

9471

9472 **C.14.2 Well-known URI**

9473 /oic/sec/sp

9474 **C.14.3 Resource type**

9475 The resource type (rt) is defined as: ['oic.r.sp'].

9476 **C.14.4 OpenAPI 2.0 definition**

```

9477 {
9478   "swagger": "2.0",
9479   "info": {
9480     "title": "Security Profile",
9481     "version": "v1.0-20190208",
9482     "license": {
9483       "name": "OCF Data Model License",
9484       "url":
9485         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9486         CENSE.md",
9487       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9488       reserved."
9489     },
9490     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9491   },
9492   "schemes": ["http"],
9493   "consumes": ["application/json"],
9494   "produces": ["application/json"],
9495   "paths": {
9496     "/oic/sec/sp" : {
9497       "get": {
9498         "description": "Resource specifying supported and active security profile(s)\n",
9499         "parameters": [
9500           {"$ref": "#/parameters/interface"}
9501         ],
9502         "responses": {
9503           "200": {
9504             "description": "",
9505             "x-example":
9506               {
9507                 "rt": ["oic.r.sp"],
9508                 "supportedprofiles" : ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9509                 "currentprofile" : "1.3.6.1.4.1.51414.0.0.1.0"
9510               },
9511             "schema": { "$ref": "#/definitions/SP" }
9512           },
9513           "400": {
9514             "description": "The request is invalid."
9515           }
9516         }
9517       },
9518       "post": {
9519         "description": "Sets or updates device provisioning status data.\n",

```

```

9520     "parameters": [
9521         { "$ref": "#/parameters/interface" },
9522     ],
9523     "name": "body",
9524     "in": "body",
9525     "required": true,
9526     "schema": { "$ref": "#/definitions/SP-Update" },
9527     "x-example":
9528     {
9529         "supportedprofiles": ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9530         "currentprofile": "1.3.6.1.4.1.51414.0.0.1.0"
9531     }
9532 }
9533 ],
9534 "responses": {
9535     "200": {
9536         "description": "",
9537         "x-example":
9538         {
9539             "rt": ["oic.r.sp"],
9540             "supportedprofiles": ["1.3.6.1.4.1.51414.0.0.1.0", " 1.3.6.1.4.1.51414.0.0.2.0"],
9541             "currentprofile": "1.3.6.1.4.1.51414.0.0.1.0"
9542         },
9543         "schema": { "$ref": "#/definitions/SP" }
9544     },
9545     "400": {
9546         "description": "The request is invalid."
9547     }
9548 }
9549 }
9550 },
9551 "parameters": {
9552     "interface": {
9553         "in": "query",
9554         "name": "if",
9555         "type": "string",
9556         "enum": ["oic.if.baseline"]
9557     }
9558 },
9559 "definitions": {
9560     "SP": {
9561         "properties": {
9562             "rt": {
9563                 "description": "Resource Type of the Resource",
9564                 "items": {
9565                     "maxLength": 64,
9566                     "type": "string",
9567                     "enum": ["oic.r.sp"]
9568                 },
9569                 "minItems": 1,
9570                 "readOnly": true,
9571                 "type": "array"
9572             },
9573             "n": {
9574                 "$ref":
9575                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9576                 schema.json#/definitions/n"
9577             },
9578             "id": {
9579                 "$ref":
9580                 "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9581                 schema.json#/definitions/id"
9582             },
9583             "currentprofile": {
9584                 "description": "Security Profile currently active",
9585                 "type": "string"
9586             },
9587             "supportedprofiles": {
9588                 "description": "Array of supported Security Profiles",
9589                 "items": {
9590

```

```

9591         "type": "string"
9592     },
9593     "type": "array"
9594 },
9595 "if": {
9596     "description": "The interface set supported by this resource",
9597     "items": {
9598         "enum": [
9599             "oic.if.baseline"
9600         ],
9601         "type": "string"
9602     },
9603     "minItems": 1,
9604     "readOnly": true,
9605     "type": "array"
9606 }
9607 },
9608 "type" : "object",
9609 "required": ["supportedprofiles", "currentprofile"]
9610 },
9611 "SP-Update" : {
9612     "properties": {
9613         "currentprofile": {
9614             "description": "Security Profile currently active",
9615             "type": "string"
9616         },
9617         "supportedprofiles": {
9618             "description": "Array of supported Security Profiles",
9619             "items": {
9620                 "type": "string"
9621             },
9622             "type": "array"
9623         }
9624     },
9625     "type" : "object"
9626 }
9627 }
9628 }
9629 }
9630 }

```

9631 C.14.5 Property definition

9632 Table C.26 defines the Properties that are part of the ['oic.r.sp'] Resource Type

9633 **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	
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
n	multiple types: see schema	No	Read Write	

9634

9635 **C.14.6 CRUDN behaviour**

9636 Table C.27 defines the CRUDN operations that are supported on the ['oic.r.sp'] Resource Type

9637 **Table C.27 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.sp']**

Create	Read	Update	Delete	Notify
	get	post		observe

9638 **C.15 Token Refresh**

9639 **C.15.1 Introduction**

9640 Obtain fresh access-token using the refresh token, client should refresh access-token before it
9641 expires.

9642
9643 **C.15.2 Well-known URI**

9644 /oic/sec/tokenrefresh

9645 **C.15.3 Resource type**

9646 The resource type (rt) is defined as: ['oic.r.tokenrefresh'].

9647 **C.15.4 OpenAPI 2.0 definition**

```
9648 {
9649   "swagger": "2.0",
9650   "info": {
9651     "title": "Token Refresh",
9652     "version": "v1.0-20181001",
9653     "license": {
9654       "name": "OCF Data Model License",
9655       "url":
9656         "https://github.com/openconnectivityfoundation/core/blob/e28a9e0a92e17042ba3e83661e4c0fbce8bdc4ba/LI
9657         CENSE.md",
9658       "x-copyright": "copyright 2016-2017, 2019 Open Connectivity Foundation, Inc. All rights
9659         reserved."
9660     },
9661     "termsOfService": "https://openconnectivityfoundation.github.io/core/DISCLAIMER.md"
9662   },
9663   "schemes": ["http"],
9664   "consumes": ["application/json"],
9665   "produces": ["application/json"],
9666   "paths": {
9667     "/oic/sec/tokenrefresh" : {
9668       "post": {
9669         "description": "Obtain fresh access-token using the refresh token, client should refresh
9670         access-token before it expires.\n",
9671         "parameters": [
9672           { "$ref": "#/parameters/interface" },
9673           {
9674             "name": "body",
9675             "in": "body",
9676             "required": true,
9677             "schema": { "$ref": "#/definitions/TokenRefresh-Request" },
9678             "x-example":
9679               {
9680                 "uid" : "123e4567-e89b-12d3-a456-d6e313b71d9f",
9681                 "di" : "9cfbeb8e-5ale-4d1c-9d01-00c04fd430c8",
9682                 "refreshtoken" : "00fe4644a6fbe5324eec"
9683               }
9684           }
9685         ],
9686         "responses": {
9687           "204": {
9688             "description": "2.04 Changed respond with new access-token\n",
9689             "x-example":
9690               {
```

```

9691         "rt": ["oic.r.tokenrefresh"],
9692         "accesstoken" : "8ce598980761869837be",
9693         "refresh token" : "d4922312b6df0518e146",
9694         "expiresin" : 3600
9695     }
9696     ,
9697     "schema": { "$ref": "#/definitions/TokenRefresh-Response" }
9698 }
9699 }
9700 }
9701 }
9702 },
9703 "parameters": {
9704     "interface" : {
9705         "in" : "query",
9706         "name" : "if",
9707         "type" : "string",
9708         "enum" : ["oic.if.baseline"]
9709     }
9710 },
9711 "definitions": {
9712     "TokenRefresh-Request" : {
9713         "properties": {
9714             "refresh token": {
9715                 "description": "Refresh token received by account management or during token refresh
9716 procedure",
9717                 "pattern": "(?!$|\\s+).*",
9718                 "type": "string"
9719             },
9720             "uid": {
9721                 "description": "Format pattern according to IETF RFC 4122.",
9722                 "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}$",
9723                 "type": "string"
9724             },
9725             "di": {
9726                 "description": "Format pattern according to IETF RFC 4122.",
9727                 "pattern": "^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$",
9728                 "type": "string"
9729             }
9730         },
9731         "type" : "object",
9732         "required": ["uid", "di", "refresh token"]
9733     },
9734     "TokenRefresh-Response" : {
9735         "properties": {
9736             "expiresin": {
9737                 "description": "Access-Token life time in seconds (-1 if permanent)",
9738                 "readOnly": true,
9739                 "type": "integer"
9740             },
9741             "rt": {
9742                 "description": "Resource Type of the Resource",
9743                 "items": {
9744                     "maxLength": 64,
9745                     "type": "string",
9746                     "enum": ["oic.r.tokenrefresh"]
9747                 },
9748                 "minItems": 1,
9749                 "readOnly": true,
9750                 "type": "array"
9751             },
9752             "refresh token": {
9753                 "description": "Refresh token received by account management or during token refresh
9754 procedure",
9755                 "pattern": "(?!$|\\s+).*",
9756                 "type": "string"
9757             },
9758             "accesstoken": {
9759                 "description": "Granted Access-Token",
9760                 "pattern": "(?!$|\\s+).*",
9761                 "readOnly": true,

```

```

9762         "type": "string"
9763     },
9764     "n": {
9765         "$ref":
9766         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9767         schema.json#/definitions/n"
9768     },
9769     "id": {
9770         "$ref":
9771         "https://openconnectivityfoundation.github.io/core/schemas/oic.common.properties.core-
9772         schema.json#/definitions/id"
9773     },
9774     "if" :
9775     {
9776         "description": "The interface set supported by this resource",
9777         "items": {
9778             "enum": [
9779                 "oic.if.baseline"
9780             ],
9781             "type": "string"
9782         },
9783         "minItems": 1,
9784         "readOnly": true,
9785         "type": "array"
9786     }
9787 },
9788 "type" : "object",
9789 "required": ["accesstoken", "refresh token", "expiresin"]
9790 }
9791 }
9792 }
9793

```

9794 C.15.5 Property definition

9795 Table C.28 defines the Properties that are part of the ['oic.r.tokenrefresh'] Resource Type

9796 **Table C.28 – The Property definitions of the Resource with type 'rt' = ['oic.r.tokenrefresh']**

Property name	Value type	Mandatory	Access mode	Description
accesstoken	string	Yes	Read Only	Granted Access-Token
n	multiple types: see schema	No	Read Write	
refresh token	string	Yes	Read Write	Refresh token received by account management or during token refresh procedure
id	multiple types: see schema	No	Read Write	
rt	array: see schema	No	Read Only	Resource Type of the Resource
expiresin	integer	Yes	Read Only	Access-Token life time in seconds (-1 if permanent)
if	array: see schema	No	Read Only	The OCF Interface set supported by this resource
refresh token	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.

9797 **C.15.6 CRUDN behaviour**

9798 Table C.29 defines the CRUDN operations that are supported on the ['oic.r.tokenrefresh']
 9799 Resource Type

9800 **Table C.29 – The CRUDN operations of the Resource with type 'rt' = ['oic.r.tokenrefresh']**

Create	Read	Update	Delete	Notify
		post		

9801

9802

9803

9804
9805
9806
9807

Annex D (informative)

OID definitions

9808 This annex captures the OIDs defined throughout the document. The OIDs listed are intended to
9809 be used within the context of an X.509 v3 certificate. MAX is an upper bound for SEQUENCES of
9810 UTF8Strings and OBJECT IDENTIFIERS and should not exceed 255.

```
9811 id-OCF OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1)
9812     private(4) enterprise(1) OCF(51414) }
9813
9814 -- OCF Security specific OIDs
9815
9816 id-ocfSecurity OBJECT IDENTIFIER ::= { id-OCF 0 }
9817 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9818
9819 -- OCF Security Categories
9820
9821 id-ocfSecurityProfile ::= { id-ocfSecurity 0 }
9822 id-ocfCertificatePolicy ::= { id-ocfSecurity 1 }
9823
9824 -- OCF Security Profiles
9825
9826 sp-undefined ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 0 }
9827 sp-baseline ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 1 }
9828 sp-black ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 2 }
9829 sp-blue ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 3 }
9830 sp-purple ::= OBJECT IDENTIFIER { id-ocfSecurityProfile 4 }
9831
9832 sp-undefined-v0 ::= ocfSecurityProfileOID (id-sp-undefined 0)
9833 sp-baseline-v0 ::= ocfSecurityProfileOID {id-sp-baseline 0}
9834 sp-black-v0 ::= ocfSecurityProfileOID {id-sp-black 0}
9835 sp-blue-v0 ::= ocfSecurityProfileOID {id-sp-blue 0}
9836 sp-purple-v0 ::= ocfSecurityProfileOID {id-sp-purple 0}
9837
9838 ocfSecurityProfileOID ::= UTF8String
9839
9840 -- OCF Security Certificate Policies
9841
9842 ocfCertificatePolicy-v1 ::= { id-ocfCertificatePolicy 2}
9843
9844 -- OCF X.509v3 Extensions
9845
9846 id-ocfX509Extensions OBJECT IDENTIFIER ::= { id-OCF 1 }
9847 id-ocfCompliance OBJECT IDENTIFIER ::= { id-ocfX509Extensions 0 }
9848 id-ocfSecurityClaims OBJECT IDENTIFIER ::= { id-ocfX509Extensions 1 }
9849 id-ocfCPLAttributes OBJECT IDENTIFIER ::= { id-ocfX509Extensions 2 }
9850
9851 ocfVersion ::= SEQUENCE {
9852     major    INTEGER,
9853     minor    INTEGER,
9854     build    INTEGER}
9855
9856 ocfCompliance ::= SEQUENCE {
9857     version          ocfVersion,
9858     securityProfile SEQUENCE SIZE (1..MAX) OF ocfSecurityProfileOID,
9859     deviceName      UTF8String,
9860     deviceManufacturer UTF8String}
9861
9862 claim-secure-boot ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 0 }
```

```
9863 claim-hw-backed-cred-storage ::= ocfSecurityClaimsOID { id-ocfSecurityClaims 1 }
9864
9865 ocfSecurityClaimsOID ::= OBJECT IDENTIFIER
9866
9867 ocfSecurityClaims ::= SEQUENCE SIZE (1..MAX) of ocfSecurityClaimsOID
9868
9869 cpl-at-IANAPen ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 0 }
9870 cpl-at-model ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 1 }
9871 cpl-at-version ::= OBJECT IDENTIFIER { id-ocfCPLAttributes 2 }
9872
9873 ocfCPLAttributes ::= SEQUENCE {
9874     cpl-at-IANAPen UTF8String,
9875     cpl-at-model UTF8String,
9876     cpl-at-version UTF8String}
```

Annex E
(informative)

Security considerations specific to Bridged Protocols

9877
9878
9879
9880

9881 The text in this Annex is provided for information only. This Annex has no normative impact. This
9882 information is applicable at the time of initial publication and may become out of date.

E.1 Security Considerations specific to the AllJoyn Protocol

9883
9884

This clause intentionally left empty.

E.2 Security Considerations specific to the Bluetooth LE Protocol

9885

9886 BLE GAP supports two security modes, security mode 1 and security mode 2. Each security
9887 mode has several security levels (see Table E.1)

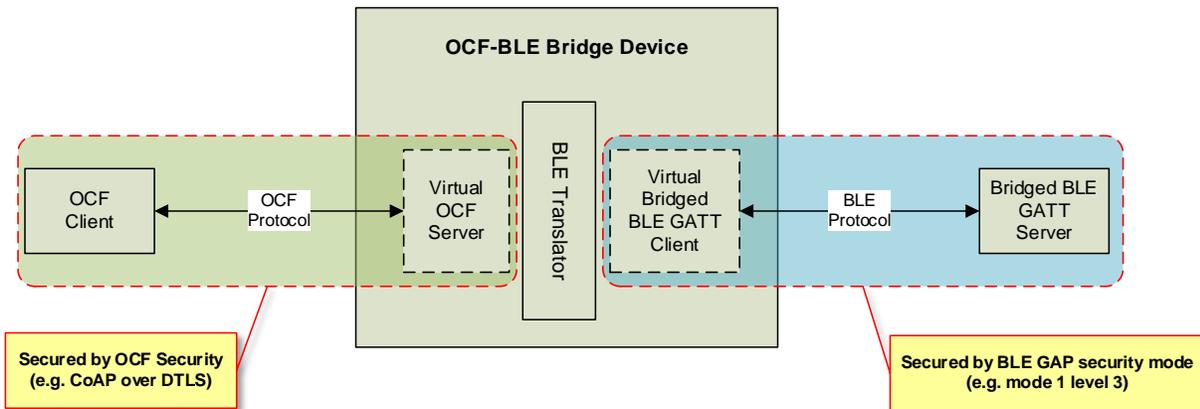
9888 Security mode 1 and Security level 2 or higher would typically be considered secure from an OCF
9889 perspective. The appropriate selection of security mode and level is left to the vendor.

Table E.1 GAP security mode

9890

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)

9891 Figure E-1 shows how communications in both ecosystems of OCF-BLE Bridge device are
9892 secured by their own security.



9893
9894

Figure E-1 Security Considerations for BLE Bridge

E.3 Security Considerations specific to the oneM2M Protocol

9895

This clause intentionally left empty.

9896

E.4 Security Considerations specific to the U+ Protocol

9897

9898 A U+ server supports one of the TLS 1.2 cipher suites as in Table E.2 defined in IETF RFC 5246.

9898

Table E.2 TLS 1.2 Cipher Suites used by U+

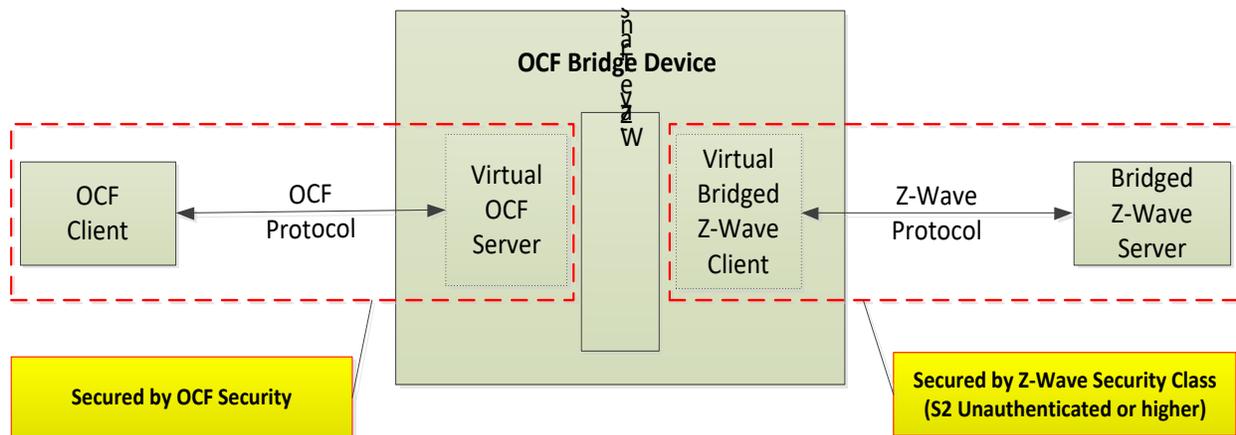
Cipher Suite
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CCM
TLS_RSA_WITH_AES_256_CCM_8
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
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

9900 The security of the Haier U+ Protocol is proprietary, and further details are presently unavailable.

9901 **E.5 Security Considerations specific to the Z-Wave Protocol**

9902 Z-Wave currently supports two kinds of security class which are S0 Security Class and S2
 9903 Security Class, as shown in Table 7 below.. Bridged Z-wave Servers using S2 Security Class for
 9904 communication with a Virtual Bridged Client would typically be considered secure from an OCF
 9905 perspective. The appropriate selection for S2 Security Class and Class Name is left to the vendor.

9906 Figure E-2 presents how OCF Client and Bridged Z-Wave Server communicate based upon their
 9907 own security.



9908

9909

Figure E-2 Security Considerations for Z-Wave Bridge

9910

All 3 types of S2 Security Class such as S2 Access Control, S2 Authenticated and S2 Unauthenticated provides the following advantages from the security perspective;

9911

9912
9913

- The unique device specific key for every secure device enables validation of device identity and prevents man-in-the-middle compromises to security

9914
9915

- The Secure cryptographic key exchange methods during inclusion achieves high level of security between the Virtual Z-Wave Client and the Bridged Z-Wave Server.

9916
9917

- Out of band key exchange for product authentication which is combined with device specific key prevents eavesdropping and man-in-the-middle attack vectors.

9918 See Table E.3 for a summary of Z-Wave Security Classes.

9919

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

9920
9921
9922
9923

On the other hand, S0 Security Class has the vulnerability of security during inclusion by exchanging of temporary 'well-known key' (e.g. 1234). As a result of that, it could lead the disclosure of the network key if the log of key exchange methods is captured, so Z-Wave devices might be no longer secure in that case.

E.6 Security Considerations specific to the Zigbee Protocol

9925
9926
9927
9928

The Zigbee 3.0 stack supports multiple security levels. A security level is supported by both the network (NWK) layer and application support (APS) layer. A security attribute in the Zigbee 3.0 stack, nwkSecurityLevel, represents the security level of a device. Further details can be found in the .

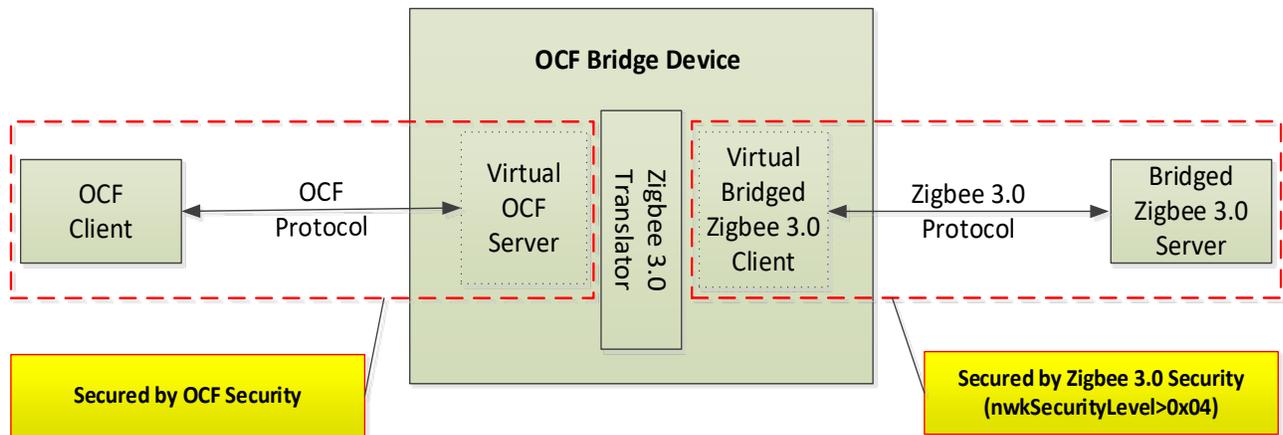
9929 The security level nwkSecurityLevel > 0x04 provides message integrity code (MIC) and/or
 9930 AES128-CCM encryption (ENC). Zigbee Servers using nwkSecurityLevel > 0x04 would typically
 9931 be considered secure from an OCF perspective. The appropriate selection for nwkSecurityLevel
 9932 is left to the vendor.

9933 See Table E.4 for a summary of the Zigbee Security Levels.

9934 **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)

9935 Figure E-3 shows how communications in both ecosystems of OCF Bridge Device are secured by
 9936 their own security.



9937
 9938

9939 **Figure E-3 Security Considerations for Zigbee Bridge**